Introdução à Web Semântica

A inteligência da informação

Casa do Código

DIEGO EIS

Sumário

- ISBN
- Agradecimentos
- Prefácio
- Sobre o livro
- 1. Introdução
- 2. Como organizamos informação?
- 3. O que é Web Semântica?
- 4. Linked Data: o conceito de dados conectados
- 5. Unicode e URI
- 6. Vocabulários
- 7. Triplas e grafos
- 8. RDFa
- 9. JSON-LD
- 10. Extra: Turtle e SPARQL
- 11. Polêmica: código limpo ainda é necessário?
- 12. Concluindo

ISBN

Impresso e PDF: 978-85-94188-06-9

EPUB: 978-85-94188-07-6

MOBI: 978-85-94188-08-3

Você pode discutir sobre este livro no Fórum da Casa do Código: http://forum.casadocodigo.com.br/.

Caso você deseje submeter alguma errata ou sugestão, acesse http://erratas.casadocodigo.com.br.

Agradecimentos

Um livro não é escrito por apenas uma pessoa, mas com o conhecimento e a ajuda de várias outras. Posso não ser a pessoa mais experiente para escrever sozinho um livro sobre Web Semântica, mas contei com o apoio de várias pessoas que me auxiliaram a escrever este livro, seja revisando o texto ou dando pitacos sobre a parte técnica.

Obrigado ao Pablo Silva e ao Willian PotHix, por terem disponibilizado o tempo sagrado de vocês para enviar suas observações, revisões, dicas e críticas sobre todo o texto escrito. Valeu!

Muito obrigado, Newton Calegari, por ter aceitado o convite de escrever o prefácio deste livro e também pelos papos sobre esse assunto tão confuso e, muitas vezes, utópico que é a Web Semântica.

Obrigado ao Manu Sporny, por ter disponibilizado tanto conteúdo importante na internet sobre JSON-LD, Web Semântica e, claro, por ter respondido meus e-mails. ;-)

Outras pessoas me ajudaram muito de forma indireta, como é o caso do Ícaro Medeiros e do Luis Cipriani. Obrigado por todo o conteúdo que vocês publicaram sobre o assunto.

Agradeço também a Marcela, minha mulher, por ter sido paciente quando fiquei ausente aos domingos e horas depois do expediente. Sem a sua paciência, certamente eu não teria conseguido. <3

Prefácio

Por muito tempo, Web Semântica foi um assunto discutido com maior força no ambiente acadêmico. Um fato que observamos em relação a isso é o grande número de conferências - de caráter acadêmico - abordando este tema. Notamos uma diferença em relação ao número de conferências relacionadas com desenvolvimento web, como por exemplo, eventos que abordam JavaScript, CSS e HTML, os quais são mais recorrentes e possuem participação ativa da comunidade.

Essa característica acadêmica vem se alterando nos últimos anos, principalmente com o aparecimento de aplicações que utilizam tecnologias de Web Semântica em diferentes domínios.

Antes de mergulhar nas questões técnicas, este livro traz uma base histórica relacionada à organização das informações e assuntos tratados também pela biblioteconomia e ciência da informação. Web Semântica é um objeto de estudo tanto dessas áreas quanto da ciência da computação, sendo muito relevante para o desenvolvimento web.

Para lidar com o desafio de relacionar e organizar a informação, Diego resgata as ideias de Vannevar Bush, com o Memex (o conceito de Hipertexto, por Ted Nelson), até o surgimento da Web (com Tim Berners-Lee).

Para explicar o que é Web Semântica, ele começa alertando o leitor para evitar a confusão entre código semântico. Isso é algo bastante comum quando estudamos linguagens de marcação (como HTML) e Web Semântica - que surgiu a partir de uma proposta de Tim Berners-Lee em 2001 - que é apontada como "uma extensão da Web atual". Escrevo entre aspas porque a "Web atual" está sempre evoluindo e possui diferentes aspectos, então, considere, nesse momento, a Web atual como "Web de Documentos".

Ainda como base para explicação do assunto principal do livro, existe a explicação sobre as fases e mudanças que ocorreram na Web, até chegar na Web de Dados.

O conjunto de ferramentas, padrões e tecnologias relacionadas à Web Semântica contribui para que as máquinas e aplicações possam processar, interpretar e realizar inferências sobre dados estruturados publicados na Web, assim como fazemos consciente e inconscientemente ao ler os mais diversos conteúdos disponíveis.

Não seria possível discutir Web Semântica sem falar sobre *linked data*, uma abordagem para publicação de dados na Web, que permite a conexão entre esses dados. Seu objetivo é enriquecer o conteúdo publicado e consumido por aplicações. Juntamente com esse tópico, o autor traz também o conceito de dados abertos, explicando de maneira clara o que eles são, como e onde podem ser usados e como contribuir para o ecossistema.

Fortemente relacionados com esses dois assuntos, estão as 5 estrelas dos *Dados Abertos Conectados* (do inglês, *Linked Open Data*), que nos mostram os diferentes níveis utilizados para classificar publicação de dados na Web. Quanto mais alto o nível, melhor será a contribuição para a Web de Dados.

O documento *Data on the Web Best Practices*, uma recomendação W3C (*World Wide Web Consortium*), apresenta um conjunto de boas práticas para publicar e consumir dados na Web, abordando aspectos mais básico, como a publicação de metadados, até tópicos de mais alto nível, como a manutenção e documentação de APIs. Ao aplicar as boas práticas recomendadas, busca-se melhorar a interação e comunicação entre os grupos que consomem e publicam dados na Web.

Como ponto de partida para descrição dos detalhes técnicos relacionados com Web Semântica, o autor apresenta a composição da sua pilha tecnológica. Nesse ponto, são esclarecidos todos os aspectos fundamentais, e isso também direciona para uma

abordagem mais prática do assunto. Ele não se prende aos conceitos teóricos sobre RDF e ontologias, mas indica para o uso de tecnologias como JSON-LD e RDFa.

Os conceitos de Unicode e URI (*Uniform Resource Identifier*) estão na base da pilha tecnológica da Web Semântica. O capítulo *Unicode e URI* detalha todos esses conceitos, explicando o comportamento das URIs quando acionadas no browser, e a sequência de passos que um navegador web faz até levar o usuário ao destino solicitado (momento em que URI se torna URL). Aqui também é mostrada a importância do uso de Unicode como mecanismo de internacionalização, explicando com riqueza de detalhes sobre as tabelas de caracteres e seu uso na internet.

Avançando com os conceitos presentes na pilha tecnológica, estão as explicações sobre vocabulários e estes são aplicados e utilizados em Web Semântica. Para contextualizar esse tópico, o autor resgatou características de como Aristóteles e Platão organizavam a informação na Grécia antiga, o que tornou as explicações sobre os vocabulários e as ontologias mais claras.

Utilizamos vocabulário o tempo todo, geralmente com o intuito de alinhar os diálogos entre pessoas, mas também podemos usá-los para alinhar o diálogo entre máquinas. Aplicamos vocabulários para descrever os dados a fim de adicionar contexto e torná-los compreensíveis pela máquina. Existem inúmeros vocabulários disponíveis para uso na Web hoje, e talvez um dos mais conhecidos seja o Schema.org, que possui termos sobre diferentes domínios.

Em uma conversa que tive com o Dan Brickley, criador do Schema.org, durante uma reunião sobre vocabulários e descrições de dados, realizada pelo W3C em Amsterdã, falamos sobre o aumento no número de desenvolvedores publicando dados estruturados em suas páginas e concordamos que isso pode ter sido impulsionado pelo fato de adotarmos maneiras pragmáticas e simples de se adicionar semântica aos nossos sites.

Comento isso porque vejo que pessoas como o Diego, que encaram a função de facilitar o acesso aos materiais relacionados à Web Semântica, são fundamentais para que mais desenvolvedores conheçam sobre o assunto.

Partindo para o conceito mais representativo de Web Semântica, há um capítulo dedicado para explicar triplas e grafos. As triplas formam a unidade mais básica desse universo, sendo formadas por "sujeito", "predicado" e "objeto", e são capazes de representar qualquer informação seguindo essa estrutura. Muitas triplas reunidas permitem a formação dos grafos, uma estrutura que mostra a conexão entre os conceitos, as coisas e os recursos.

Para facilitar o entendimento do conceito de grafos e redes na Web Semântica, vemos no livro exemplos de grafos representados com RDFa, Turtle e N-Triples, linguagens de serialização RDF. O capítulo *RDFa* é dedicado a essa linguagem, que traz atributos específicos para adicionar conteúdo RDF em documentos HTML. Diego faz uma explicação incremental mostrando exemplos de trechos de código HTML com atributos RDFa e utilizando o vocabulário Schema.org para a descrição dos recursos. Nesse momento, ficará nítida a facilidade de adicionar semântica aos projetos e à rotina de desenvolvimento web.

O capítulo *JSON-LD* inicia mostrando o valor e a simplicidade do formato JSON (*JavaScript Object Notation*) com a sintaxe de chave e valor. A simplicidade inerente do JSON o tornou um importante formato para intercâmbio de dados na Web. Se analisarmos em uma escala, arrisco a dizer que a sua simplicidade está em um extremo, enquanto a complexidade presente em RDF/XML está em outro.

Levando em conta essa simplicidade e facilidade no uso do JSON, é importante conhecer sobre JSON-LD (*JSON for Linked Data*), formato que possui as mesmas características do JSON, contudo permite representar triplas e grafos. Além disso, JSON-LD é uma recomendação W3C relativamente recente, porém, já é amplamente

usado para agregar marcações semânticas às páginas web. Existem algumas vantagens de se utilizar JSON-LD em vez de RDFa em alguns cenários, casos estes que Diego apresenta nesse capítulo.

Do mesmo modo que ocorre na seção sobre RDFa, nesta há uma explicação rica e detalhada apresentando trechos de código e exemplos, como os produtos do Google que utilizam JSON-LD. Essa explicação de modo incremental torna o entendimento muito mais fácil. Esse capítulo ensina como utilizar JSON-LD na prática.

Ao chegar até este ponto do livro, você terá adquirido uma bagagem valiosa de conhecimento relacionado à Web Semântica, mas Diego nos presenteia com um capítulo extra que auxilia aqueles que querem ir mais a fundo no assunto. O capítulo *Extra: Turtle e SPARQL* traz conteúdo sobre a linguagem de serialização Turtle, muito conhecida e usada neste contexto, e também sobre a linguagem de consultas SPARQL, uma linguagem robusta utilizada para realizar consultas em bases de dados conhecidas como *triple stores* - bases que armazenam dados em RDF.

O ponto mais marcante deste livro é a maneira pragmática com que todos os conceitos são ensinados e demonstrados, fator que contribui muito para facilitar a aplicação de Web Semântica. O Diego encarou o desafio de trazer este assunto, por vezes visto como muito complexo e pouco prático, próximo do cotidiano do desenvolvedor web. Esse foi um grande desafio, pois passa por etapas de mostrar todo o conceito e teoria que levaram ao surgimento da Web Semântica, como vem sendo aplicada na indústria, o processo de padronização e, por fim, chegar a pontos em que temos uma abordagem pragmática para utilização de Web Semântica.

Aproveite a leitura e aprenda muito com este ótimo livro!

Um abraço,

Newton Calegari

Sobre o livro

Por que decidi escrever e para quem é o livro?

Estudar as tecnologias que formam a Web Semântica é um desafio. Os assuntos são difíceis de encontrar e, quando encontramos, falta contexto para entender boa parte dos textos. Também percebi que muitos desenvolvedores têm dúvidas sobre o que é Web Semântica e qual seria o objetivo de termos uma Web com dados relacionados e abertos.

Pensando nisso, resolvi escrever este livro, explorando principalmente a parte conceitual e teórica do assunto. Você vai descobrir que a Web Semântica pode ser um assunto altamente técnico, contudo, a parte que mais impacta os desenvolvedores e nosso trabalho diário é muito simples de ser aplicada e implementada.

Por isso, deixei a parte profundamente técnica fora deste livro. Porém, espero que a história, a teoria e o conceito da Web Semântica o incentivem a estudar mais sobre o assunto.

Durante toda a leitura, ao final de cada capítulo, coloquei links de referências para que você se aprofunde nos assuntos citados. Esses links serviram como apoio para a escrita do livro e, com certeza, vão ajudá-lo nas suas pesquisas.

Não vou mentir: tentei filtrar o máximo de conteúdo inútil, mas não consegui. Se eu tivesse tido êxito, este livro teria apenas um punhado de capítulos. Há muito conteúdo por aí que você, desenvolvedor web, não precisa saber para fazer com que seu produto ou site esteja de acordo com os conceitos da Web Semântica. Mas durante todo o meu estudo, encontrei detalhes que ampliam mais seu entendimento.

Um dos conteúdos que eu filtrei foi o de RDF. É praticamente unânime falar sobre RDF quando se fala sobre Web Semântica.

Mas decidi retirar um capítulo inteiro sobre RDF do livro por um simples motivo: eu nunca usei RDF para nada e, muito provavelmente, você também não usará. Por isso, decidi deixar dois capítulos mais mão na massa ensinando RDFa e JSON-LD (meu predileto).

Discuta o livro comigo

Gostaria muito que você comentasse comigo sobre este livro no fórum da Casa do Código (http://forum.casadocodigo.com.br/). Dúvidas, indicações de materiais, erratas ou qualquer coisa relacionada à Web Semântica, você também pode entrar em contato comigo pelo Twitter (@*diegoeis*) ou por e-mail (diego@tableless.com.br). Eu também abordo esse e outros assuntos no meu site pessoal (http://diegoeis.com) ou no Tableless (http://tableless.com.br).

Capítulo 1
Introdução

> *The Semantic Web isn't just about putting data on the Web. It is about making links, so that a person or a machine can explore the Web of Data. With Linked Data, when you have some of it, you can find other, related, data.* — **Tim Berners-Lee**

A Web é um lugar fantástico. Ela mudou a forma como o mundo funciona, como as pessoas se relacionam e fazem negócios. Em um livreto chamado **A Internet como paradigma**, da Editora Expressão e Cultura, vários autores descrevem como a internet (incluindo Web) mudou a vida das pessoas. Eu tenho esse livro desde 2006 e, até hoje, a internet/Web tem causado mudanças profundas em todos os mercados existentes. Quando visualizamos a internet como plataforma, esse poder transformador se amplifica.

Eu já falei isso em muitas palestras e artigos: se eu tivesse de resumir a utilidade da Web em apenas uma palavra, essa palavra seria **compartilhar**. A internet foi criada para compartilhar informação. A Web, por sua vez, facilitou o processo se transformando em uma plataforma para que usuários consumissem informação. Não demorou muito para que os usuários ganhassem a capacidade de produzir conteúdo em blogs, redes sociais e várias outras plataformas.

Compartilhar informação é o principal objetivo da Web hoje e será seu principal objetivo amanhã. Se compartilhar é um dos principais objetivos da Web, cada vez mais precisamos criar meios para que essa tarefa seja cumprida e, principalmente, precisamos transformar essa massa enorme informação que a Web guarda em algo realmente útil. O que me leva a pergunta: o que podemos fazer com essa informação?

1.1 DIKW — Data, Information, Knowledge, Wisdom

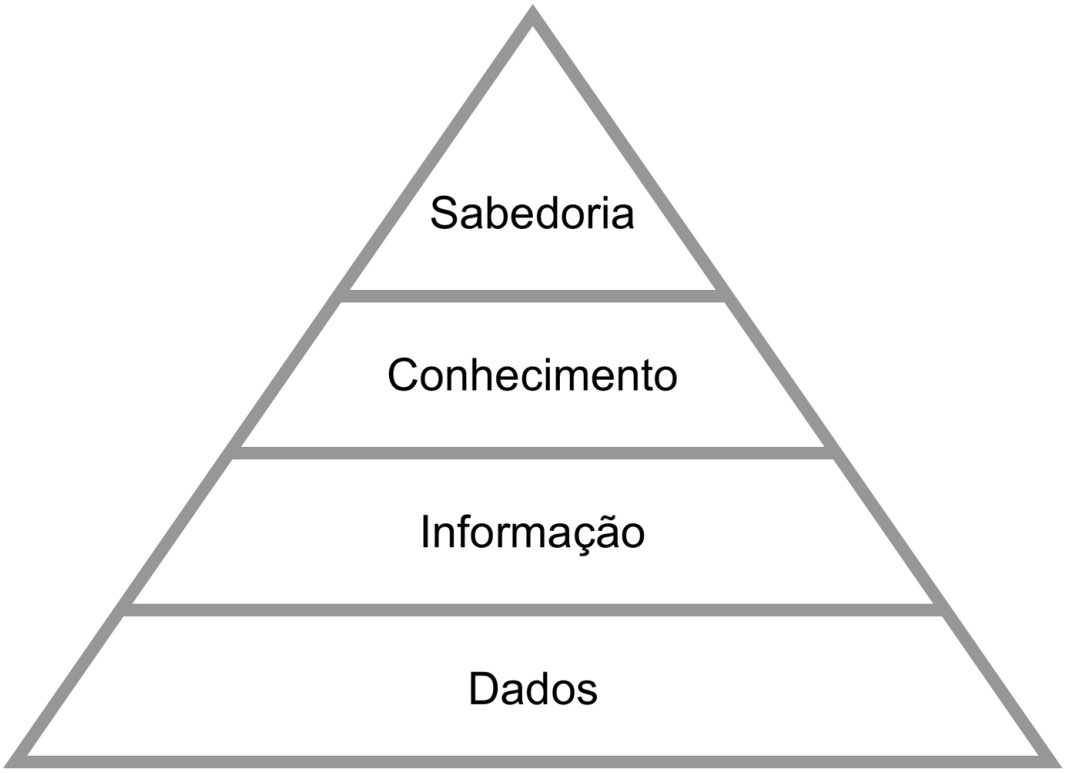
Russell Ackoff foi um dos pesquisadores que difundiram a **Pirâmide do Conhecimento**, ou a Hierarquia do Conhecimento.

> *An ounce of information is worth a pound of data. An ounce of knowledge is worth a pound of information. An ounce of understanding is worth a pound of knowledge.* — **Russel Ackoff**

O artigo de Russell, *From Data to Wisdom*, critica a forma como os alunos aprendem nas escolas do mundo todo, mostrando que a forma tradicional de aprendizado é “quebrada”. Ele diz que as escolas perdem muito tempo tentando **transmitir informação**, bem menos tempo é dedicado para **ensinar como obter conhecimento** e quase nenhum tempo é dedicado para **transmitir entendimento**.

Ou seja, quase nenhum tempo é dedicado para ensinar como **obter entendimento**. Ele diz que os alunos: *“não apenas não sabem, mas eles não sabem o que eles não sabem”.*

Nesse artigo, ele explica o processo como a informação se transforma em entendimento ou sabedoria. Do inglês, conseguimos ter a sigla DIKW, a abreviação das palavras ***D****ata*, ***I****nformation*, ***K****nowledge* e ***W****isdom*, que formam a Pirâmide do Conhecimento. Algo mais ou menos assim:



As cinco categorias estão relacionadas dessa forma:

1. **Dados** são dados concebidos na forma de símbolos ou sinais. Eles não são úteis ou relevantes até a hora do seu processamento. É fácil de entender quando pensamos em algo mais programável. Pense nos bancos de dados, onde você tem números, letras e outros caracteres que por si só não significam nada. Quando esses dados são de alguma forma processados (por um computador ou pelo seu cérebro) e relacionados, eles se transformam na próxima camada da pirâmide: a informação.

2. **Informação** é o resultado do processamento e relacionamento dos dados. A informação se refere à descrição, à perspectiva e à definição dos dados. A informação, diferente dos dados, é útil, nos trazendo respostas aos questionamentos. Alguns artigos sobre a Pirâmide do Conhecimento resumem a base para qualquer questionamento em: quem, o quê, onde, quantos e quando.

3. **Conhecimento** é o resultado da informação processada, organizada e estruturada de alguma forma que ela possa ser aplicada e transformada em ação. Conhecimento pode ser categorizado como uma **coleção de informações**. Podemos dizer que, quando memorizamos informações, nós estamos colecionando conhecimento. Mas temos de entender que juntar conhecimento por meio de memorização não é a mesma coisa que entender o que essa informação significa. Podemos dizer também que conhecimento é a capacidade de processar ou transformar uma informação, qualitativa ou quantitativa, em outra informação, em outro conhecimento ou ação. Por exemplo, seguir instruções, procedimentos de emergência, entender materiais científicos etc.
4. **Sabedoria** é lidar com valores humanos e pessoais. Envolve exercício de julgamento. Avaliações de conhecimento são baseadas em lógica, que a princípio podem ser programadas e automatizadas por computadores. Computadores que executam tarefas e automatizam são eficazes. Eficiência e eficácia são muitas vezes confundidas como se fossem a mesma coisa. Eficiência tem a ver com fazer as coisas da maneira certa. Eficácia tem a ver com fazer as coisas certas. Inteligência é a habilidade de aumentar a eficiência. Sabedoria é a habilidade de aumentar a eficácia.

> *Data is not information, information is not knowledge, knowledge is not understanding, understanding is not wisdom.* — **Clifford Stoll**

Levando em consideração que a Web guarda uma grande quantidade de dados importantes (e alguns bem inúteis também), como podemos transformar esses dados em informação, conhecimento e sabedoria? Como podemos estruturar esses dados a fim de facilitar a extração de informação? Podemos usar um formato padrão para estruturar e relacionar esses dados? Como

resolvemos a ambiguidade desses dados? Como podemos relacionar essas informações, transformando a Web em um grande banco de dados global?

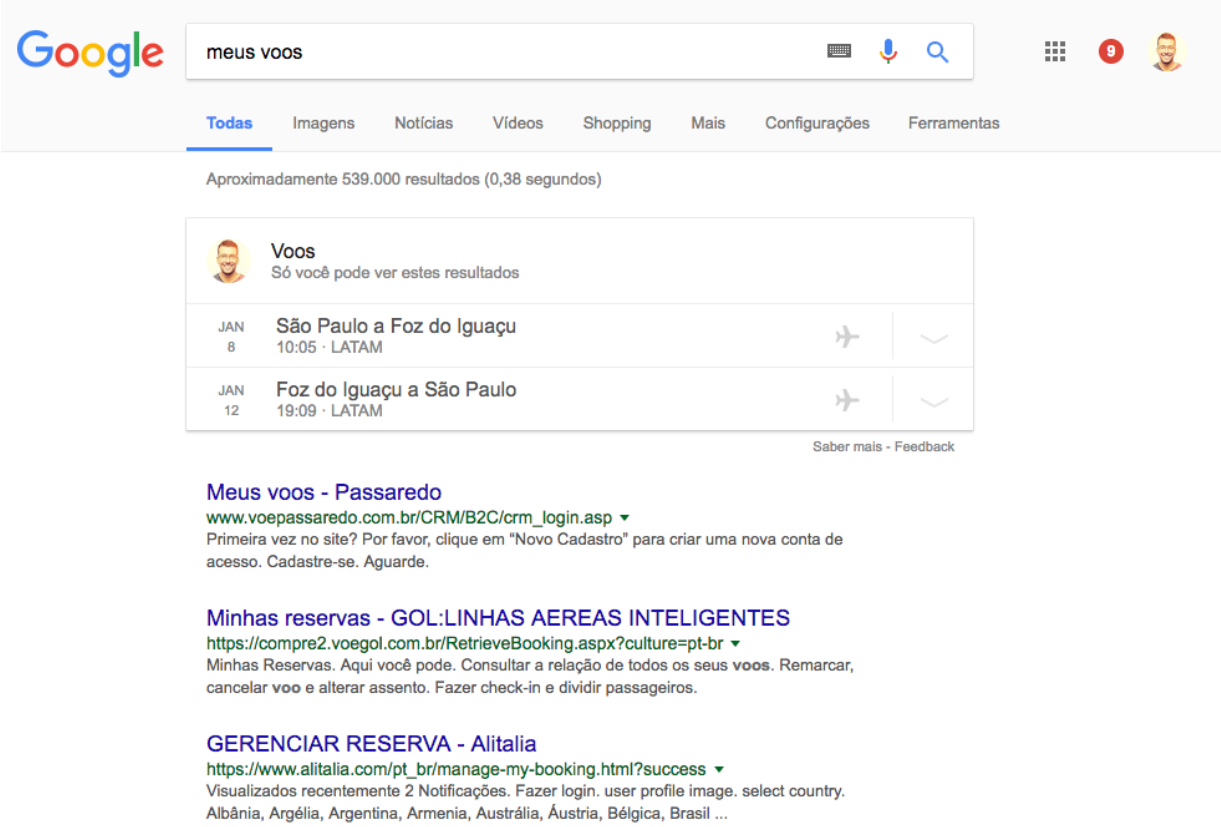
> *Semantic web is all about data integration.* — **Toby Segaran, Colin Evans e Jamie Taylor**

A Web Semântica é uma evolução da Web atual. Enquanto a Web atual significa criar e produzir informação, a Web Semântica significa integrar essas informações de forma que elas façam sentido.

Organizar e relacionar a informação da Web é uma tarefa difícil. Persuadir o mundo inteiro para que cada um possa abrir e publicar seus dados em formatos padronizados é um trabalho gigante, mas não é impossível. Do mesmo jeito que houve uma revolução quando a internet surgiu, o estabelecimento da Web Semântica também será uma mudança de paradigma.

1.2 Web Semântica já está na sua vida

Atualmente, você já tem a Web Semântica sutilmente presente na sua vida. Se você tiver viagens marcadas ou hotéis reservados, tente digitar no Google: **Meus voos** ou **Minhas reservas de hotel**. Na listagem de resultados, vai aparecer um card parecido com o seguinte:


Google
meus voos
Todas
Imagens
Notícias
Vídeos
Shopping
Mais
Configurações
Ferramentas
Aproximadamente 539.000 resultados (0,38 segundos)
Voos
Só você pode ver estes resultados
JAN 8
São Paulo a Foz do Iguaçu
10:05 · LATAM
JAN 12
Foz do Iguaçu a São Paulo
19:09 · LATAM
Saber mais - Feedback
Meus voos - Passaredo
www.voepassaredo.com.br/CRM/B2C/crm_login.asp
Primeira vez no site? Por favor, clique em "Novo Cadastro" para criar uma nova conta de acesso. Cadastre-se. Aguarde.
Minhas reservas - GOL:LINHAS AEREAS INTELIGENTES
https://compre2.voegol.com.br/RetrieveBooking.aspx?culture=pt-br
Minhas Reservas. Aqui você pode. Consultar a relação de todos os seus voos. Remarcar, cancelar voo e alterar assento. Fazer check-in e dividir passageiros.
GERENCIAR RESERVA - Alitalia
https://www.alitalia.com/pt_br/manage-my-booking.html?success
Visualizados recentemente 2 Notificações. Fazer login. user profile image. select country. Albânia, Argélia, Argentina, Armenia, Austrália, Áustria, Bélgica, Brasil ...

Versões comerciais do banco de dados Oracle já tem suporte nativo à Web Semântica. A ideia é fazer com que os nossos sistemas não entendam apenas os relacionamentos entre os nossos dados, mas também facilitem a descoberta de novas relações entre esses dados. A Oracle então serve em seus bancos de dados o que eles chamam de *capabilities data-management*.

> *Work is ongoing with C, Java, XML, SOA, database and application server technologies, and W3C standards. At the New England Development Center, our developers are familiar with W3C graph data languages, including Resource Description Framework, Web Ontology Language, and SPARQL; reasoning, including first-order logic and description logic; network analytics, visualization, and management of graph data; and domain ontologies, taxonomies, and vocabularies.* — Oracle

O Google tem feito mudanças em seu algoritmo para mantê-lo atualizado para as necessidades atuais. Em 2013, ele fez uma mudança bem drástica chamada **Hummingbird**. Não se trata de uma simples atualização do algoritmo antigo, mas uma mudança drástica para aumentar a compreensão dos significados dos termos de busca, a fim de entender e extrair termos completos e maiores, não apenas palavras isoladas.

Mudanças assim são apenas uma introdução da era da cognição e da linguagem natural. Você já vê traços da linguagem natural em assistentes como a Siri da Apple, Cortana da Microsoft ou o assistente do Google. Quando falo sobre sistemas que usam a linguagem natural para prover informações para seus usuários, quero dizer que você poderá falar com esses “assistentes digitais” da forma mais natural possível, como se estivesse conversando com outra pessoa.

Em projetos mais ousados com o Watson (https://www.ibm.com/watson/) da IBM, a Semântica está em um nível mais puro. Se você não conhece o Watson, aqui vai uma explicação bem simplista: o Watson é um sistema de processamento de linguagem natural de perguntas e respostas. Na verdade, ele é bem mais do que isso.

A IBM mesmo define o Watson como **uma tecnologia cognitiva que pensa como um ser humano**. Ele não apenas **entende** o que você fala, mas baseado em conceitos de Machine Learning, ele **aprende** o que você fala, conseguindo recuperar informações, identificar referências, gerar hipóteses e várias outras mágicas.

A IBM tem utilizado os conceitos de Linked Data e as bases de conhecimento, como a DBPedia e Geonames, para gerar respostas, além de marcar essas respostas sob vários pontos de vista, tais como coerção de tipo e proximidade geográfica.

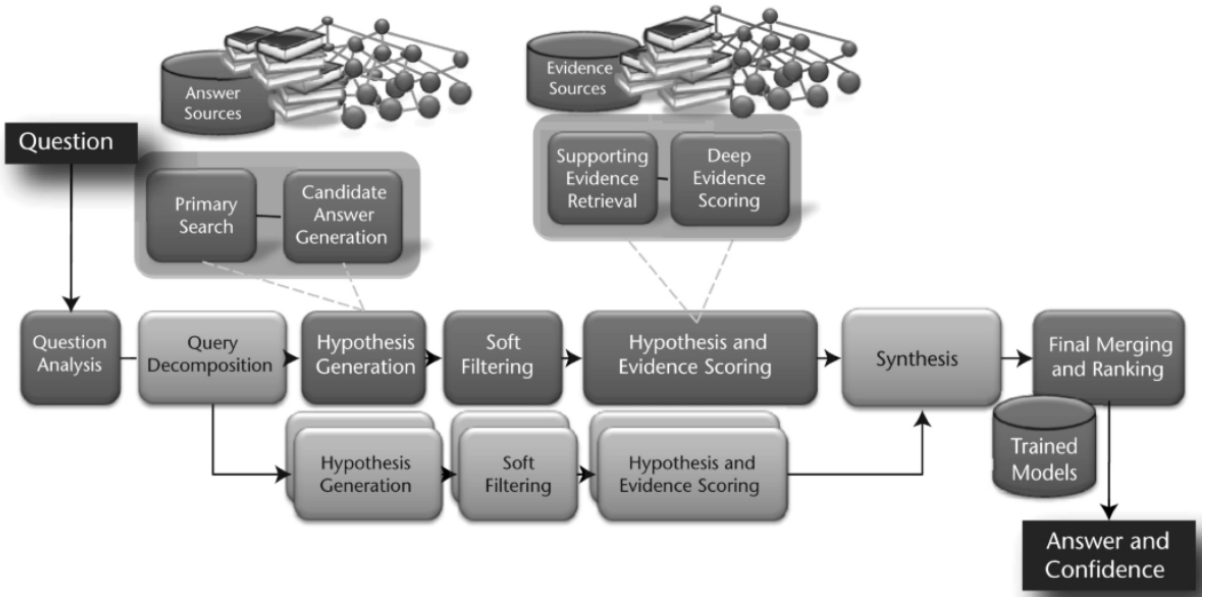


Figura 1.3: Fonte: http://www.aclweb.org/anthology/W13-3413

Imagine a combinação de uma máquina desse nível de autonomia, ligada a uma Web cujo os dados estejam totalmente relacionados. Sim, ela se chamaria Skynet. Mas esse é o poder que a Web Semântica pode alcançar. E só para constar: o Watson nasceu como um desafio de construir um sistema que pudesse ganhar de seres humanos em um jogo de perguntas e respostas muito famoso lá nos EUA, chamado *Jeopardy!*.

A Web Semântica é só um passo para um lugar muito mais espetacular e tecnológico que o nosso cenário atual. A Web será o palco de toda essa revolução. É onde as máquinas poderão encontrar informação abundante para conseguirem nos ajudar de forma inteligente e rápida.

Para iniciarmos os nossos estudos, se possível, assista essa apresentação no TED do Tim Berners-Lee. Tem apenas 16 minutos. Tomara que esse vídeo te inspire, assim como me inspirou a escrever o livro *The Next Web* (mais em: https://www.ted.com/talks/tim_berners_lee_on_the_next_web).

1.3 Para ver mais

- Documentação do W3C — http://www.w3.org/standards/semanticweb/
- From Data to Wisdom - Russel Ackoff — http://faculty.ung.edu/kmelton/Documents/DataWisdom.pdf
- Hierarquia DIKW — https://pt.wikipedia.org/wiki/Hierarquia_DIKW
- If Russ Ackoff had given TED Talk… — https://www.youtube.com/watch?v=OqEeIG8aPPk
- Systems-thinking: A Little Filme About a Big Idea — https://www.youtube.com/watch?v=-sfiReUu3o0
- Tim Berners-Lee: The next Web of open linked data — https://www.youtube.com/watch?v=OM6XIICm_qo
- Histórias sobre os Sites de Busca — https://sites.google.com/site/historiasobreossitesdebusca/a-web-semantica
- Oracle - Semantic Web Database Technologies — http://www.oracle.com/webfolder/college-recruiting/projects/semantic-web-database.html

- BELLINGER, Gene; CASTRO, Durval; MILLS, Anthony. *Data, Information, Knowledge and Wisdom.* 2004. Disponível em: http://www.systems-thinking.org/dikw/dikw.htm.

- BERNERS-LEE, Tim. *Linked Data.* 2006. Disponível em: https://www.w3.org/DesignIssues/LinkedData.

- BERNSTEIN, Jay H. *The Data-Information-Knowledge-Wisdom Hierarchy and its Antithesis.* 2009. Disponível em: https://journals.lib.washington.edu/index.php/nasko/article/viewFile/12806/11288.

- BERNSTEIN, Abraham; HENDLER, James; NOY, Natalya. *A new look at the Semantic Web.* Communications of the ACM, v. 59, n. 9, p. 35-37, 2016. Disponível em:

http://cacm.acm.org/magazines/2016/9/206254-a-new-look-at-the-semantic-web/fulltext#R3.

- GADOMSKI, Adam Maria. *Information, Preferences and Knowledge universal interrelations.* 1999. Disponível em: http://erg4146.casaccia.enea.it/wwwerg26701/gad-dict.htm.
- GLIOZZO, Alfio; BIRAN, Or; PATWARDHAN, Siddharth; MCKEOWN, Kathleen. *Semantic Technologies in IBM Watson.* Proceedings of the Fourth Workshop on Teaching Natural Language Processing, p. 85–92, August 2013. Disponível em: http://www.aclweb.org/anthology/W13-3413.
- SEGARAN, Toby; EVANS, Colin; TAYLOR, Jamie. *Programming the Semantic Web:* Build Flexible Applications with Graph Data. O'Reilly Media, 2009.
- VERBORGH, Ruben. *Web fundamentals:* The Semantic Web & Linked Data. 2017. Disponível em: http://rubenverborgh.github.io/WebFundamentals/semantic-web/.

Capítulo 2
Como organizamos informação?

> *Data helps solve problems.* — **Anne Wojcicki**

2.1 Produzindo informação como nunca

A humanidade nunca produziu tanto conteúdo como hoje. Em um artigo escrito em 2010 para o TechCrunch, Eric Schmidt (SIEGLER, 2010) citou que, a cada dois dias, a humanidade produz mais conteúdo do que produzia em 2003. Outro artigo na Science Daily (SINTEF, 2013) - site sobre ciência e tecnologia -, relata que 90% dos dados que existem no mundo foram criados nos últimos 2 anos. O artigo em questão foi publicado em 2013. Isso quer dizer que, entre 2011 e 2013, a humanidade criou 90% dos dados que existiam no mundo até aquela época. Eu não sei você, mas fico bastante impressionado com essas informações.

Gerar, ler e interpretar dados faz parte da nossa vida cotidiana, e é tão natural como dormir. O mundo é movido por dados gerados no mercado financeiro. Até a decisão de sair ou não com o seu guarda-chuva é baseado em dados meteorológicos que a moça do tempo informou no jornal da noite anterior. Você, como pessoa, gera um enorme número de dados todos os dias.

Nos últimos anos, fiquei viciado em medir alguns dados relacionados à minha saúde. Eu precisava perder peso rápido e queria conhecer mais meu corpo e meus hábitos. Para tanto, comprei um sensor (FitBit One) que me ajudou a medir a quantidade de passos que ando diariamente, qualidade do meu sono (medindo quanto tempo dormi, quantas vezes acordei durante a noite, quanto

tempo fiquei em sono profundo), quantidade de calorias ingeridas e perdidas durante o dia, quantidade de quilômetros que percorri, medição de massa magra, massa gorda, medidas do corpo e um monte de outras coisas. Com o tempo, consegui entender esses dados de forma que eu passei a me conhecer mais e isso me ajudou a mudar meu peso: saí de 117kg para 92kg (https://www.fitbit.com/user/24C4TG).

Além disso, comecei a fazer tracking de todos os dados que poderia gerar automaticamente usando um sistema chamado *Exist.io*. Nele eu consigo integrar com uma série de serviços online, a fim de juntar tudo num lugar só os dados que produzo enquanto trabalho, me exercito ou simplesmente navego na internet. A graça disso é que ele me dá análises de correlação entre esses dados, mostrando-me insights bem interessantes sobre meus hábitos.

Na internet, nós geramos dados o tempo inteiro, de forma consciente e inconsciente. Toda vez que entramos em um site, geramos dados para o dono daquele site como número de páginas visitadas, tempo de permanência em cada página, links clicados, qual o seu browser, sistema operacional, tamanho do monitor etc. Todos esses dados, depois de processados, geram uma massa de informações que ajudarão o dono do site a tomar decisões importantes.

Você gera dados de forma muito consciente quando interage nas redes sociais: são dados em forma de textos, imagem, som, vídeo. Até quando você dá um like, apertando o coraçãozinho do Twitter, um dado é gerado e vai ajudar a rede social entender melhor seu perfil. A internet guarda dados de empresas, governo, científicos, dados sobre acontecimentos históricos ou notícias de última hora, e principalmente dados pessoais. Você pode citar uma série de outros exemplos aqui. Mas existe um problema: a grande maioria desses dados não pode ser relacionada, conectada e reutilizada.

Em 1945, quando Vannevar Bush escreveu *As We May Think* (1945), um dos seus questionamentos era que a humanidade

produzia conteúdo a todo momento, mas esse conteúdo não podia ser consultado de maneira fácil, e principalmente, não podia ser relacionado. Relacionar os dados é um passo para você transformar todos esses bits e bytes em algo mais importante: informação. Mas como fazemos isso?

2.2 Transformando dados em informação

A palavra **dado** vem de uma única palavra latina **datum**, que significa "algo dado". Com o passar dos anos, esse significado mudou um bocado e hoje a palavra "dado" é como o plural de *datum*.

Mas o que é dado? Dados são a matéria prima da informação. Um dado, sozinho, raramente traz algum sentido ou significado de forma que possamos usá-lo de forma útil. Dados são simplesmente fatos de pedaços de informação, mas não informação em si. Dado é algo cru, que precisa ser processado, organizado, interpretado ou estruturado para que possamos extrair algo realmente útil e que tenha significado, resultando em informação. Informação é o resultado do tratamento e do relacionamento de dados.

Dados, por si só, não são nada. Mesmo assim, você precisa extrair esses dados de uma fonte segura, fiel, para que eles sejam realmente fatos e não algo infundado. Dados se tornam mais poderosos quando os relacionamos com outros dados.

Em uma apresentação no TED, Hans Rosling (2006) mostra como os dados podem nos dar informações importantes sobre o desenvolvimento humano. Imagine se todos os órgãos governamentais, ONGs e outras instituições disponibilizassem seus dados para que pesquisadores como Hans Rosling pudessem organizar e extrair informações importantes para o mundo inteiro. O próprio Hans comenta em seu vídeo que os dados mais importantes

estão presos em bancos de dados, sendo vendidos em vez de disponibilizados de graça, em formatos incompatíveis em vez de serem acessíveis por qualquer um.

Mas suponha que temos os dados de uma fonte segura, que possamos tratá-los, interpretá-los e estruturá-los de forma que pudéssemos extrair facilmente essas informações, como você organizaria essa massa gigante de dados?

2.3 Como organizamos informação?

Primeiro, por que precisamos organizar a informação? A resposta é simples: para facilitar a consulta dessa informação. Se geramos informação a partir de dados, essa informação será usada para algum objetivo. Toda a informação produzida precisa ser consultada, seja por você, um ser humano ou por máquinas.

Quando digo máquinas, quero dizer qualquer coisa que possa ler essas informações e reutilizá-las para alcançar algum objetivo. Por exemplo: os sistemas de busca como o Google ou qualquer outro tipo de sistema que possa querer indexar as informações disponíveis na internet.

> *A record if it is to be useful to science, must be continuously extended, it must be stored, and above all it must be consulted.*
> — **Vannevar Bush**

A produção desenfreada de informação não é um problema novo. Pesquisadores e estudiosos já discutiam maneiras de guardar informação de forma que ela pudesse ser consultada e, principalmente, relacionada a qualquer momento. Em 1945, era comum guardar informação em microfilmes, fitas magnéticas ou em antigos discos de cera. Só em 1982 os CDs foram lançados pela

Philips e pela Sony. Em 1996, sendo um adolescente, eu já tinha acesso a HDs de 3GB. E daí para a frente, guardar informação não era mais um problema sério.

Mas só guardar informação não é o suficiente. Precisamos ser capazes de encontrar a informação quando precisarmos dela. Deve ser possível relacionar diferentes formatos de informações. Logo, a forma como se organiza a informação é importante.

Existem diversas maneiras de organizar informação, alguns exemplos: por localização, alfabeticamente, tempo, categorização e hierarquicamente. Essas são as maneias que podemos usar na vida real, com coisas físicas, por exemplo, organizando livros, documentos ou diagramas. Com a evolução da tecnologia, nós conseguimos organizar as informações de forma mais abrangente, como vemos mais a seguir.

Alfabeticamente

Essa forma de organizar é bem comum. Vale para organizar desde uma lista simples de palavras até um dicionário. É muito útil quando a pessoa conhece o termo a ser pesquisado. Assim ela consegue encontrar facilmente a informação. Mas se você não conhece o termo ou a palavra que procura, esse formato não é tão simples.

Se você trata de vários termos de assuntos diferentes, por exemplo, fica complicado para a pessoa procurar, referenciar e relacionar toda a informação de um mesmo assunto, quando se está organizado alfabeticamente. Para organizar, por exemplo, livros pelos nomes dos autores, é uma boa forma de organização, mas para livros cujos termos trabalham se referenciando uns aos outros, não é algo fácil de se entender.

Por tempo

Uma forma bastante comum é organizar a informação de forma cronológica. Essa forma é muito comum quando organizamos

informações históricas, porque conseguimos mostrar de forma linear os acontecimentos.

Essa organização é muito comum em blogs, em que organizamos as postagens de forma que o leitor tenha acesso às postagens mais novas, para as mais antigas. O Twitter, até pouco tempo, organizava sua timeline dessa forma, mostrando os tweets mais novos e depois os mais antigos. Aliás, esse formato era muito usado por quase todas as redes sociais. Só depois que algumas das redes sociais piraram e começaram a criar algoritmos malucos para tentar organizar as atualizações da sua timeline de forma mais relevante, baseando-se em diversos fatores comportamentais dos usuários.

Categorização

Esse formato é muito visto em websites, principalmente em e-commerces que organizam seus produtos dentro de categorias. Essa forma é muito boa para que você consiga encontrar uma informação sempre que necessário.

Você não precisa decorar o modelo de um produto, por exemplo, para encontrá-lo novamente. Basta navegar até a sua categoria e procurá-lo em uma lista.

Mas falando da vida real, esse tipo de organização é visto em supermercados, lojas de roupa, lojas de música (ainda existe?) etc.

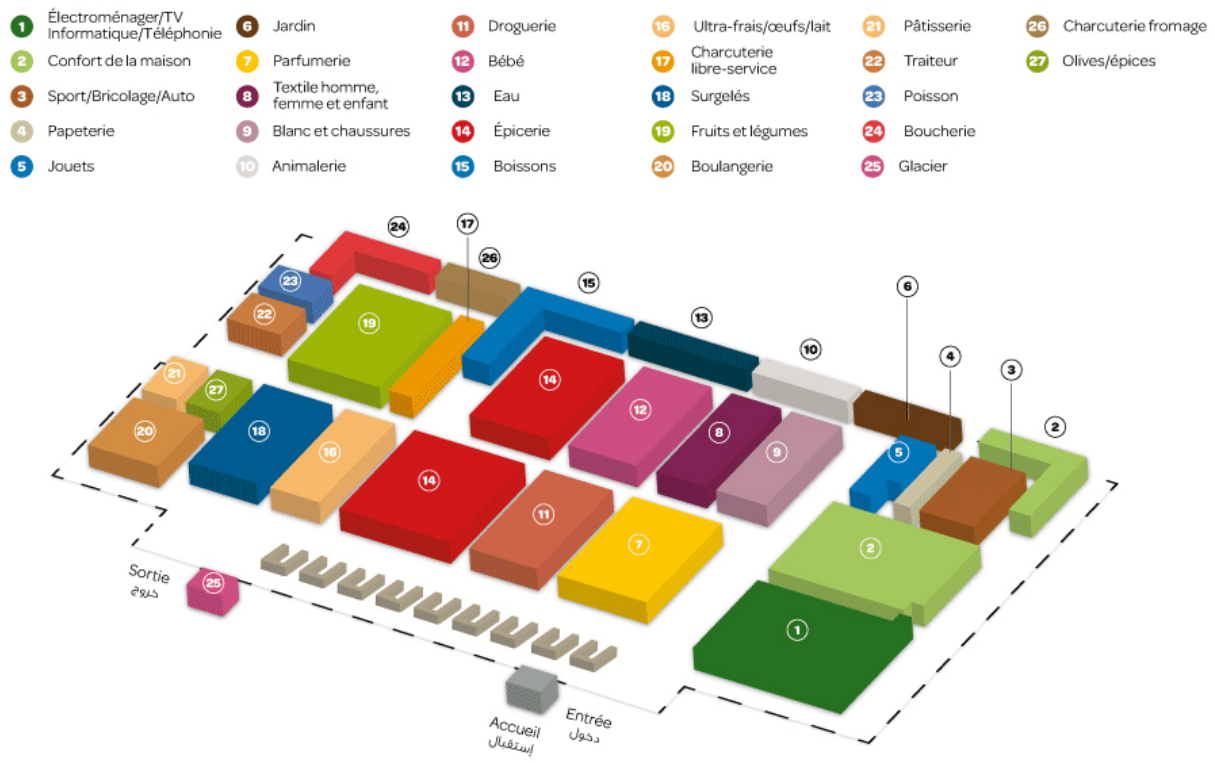


Figura 2.1: Fonte: http://www.carrefour.dz/mon-magasin/

Curiosidade

Em supermercados, pelo menos nos EUA, há algumas regras para organizar os produtos nas prateleiras: as prateleiras de cima guardam os produtos premium, dependendo da categoria, ficam os produtos mais saudáveis ou produtos locais. Já na segunda ou terceira prateleira, que é onde os magnatas dos supermercados chamam de *Bull's-Eye Zone*, ficam as marcas mais conhecidas e as que vendem mais. Geralmente, são as marcas que têm mais dinheiro para posicionar seus produtos nos lugares mais privilegiados.

Nas últimas prateleiras, geralmente ficam os produtos de fabricação própria do supermercado, ou produtos que fazem sucesso com as crianças. Normalmente, os produtos das últimas prateleiras têm os preços mais acessíveis.

Figura 2.2: Fonte: http://bit.ly/grocery-store-shelf

Categorização é uma das formas mais efetivas de organizar informação exatamente porque você não precisa memorizar muita coisa para poder recuperar essa informação quando for necessário. É um formato mais amigável.

Por localização

Quando organizamos por localização, geralmente usamos uma ilustração ou um diagrama para ajudar. Imagine um mapa, ou uma ilustração que mostra o que é e onde cada coisa está localizada. Tome como exemplo esse diagrama do coração:

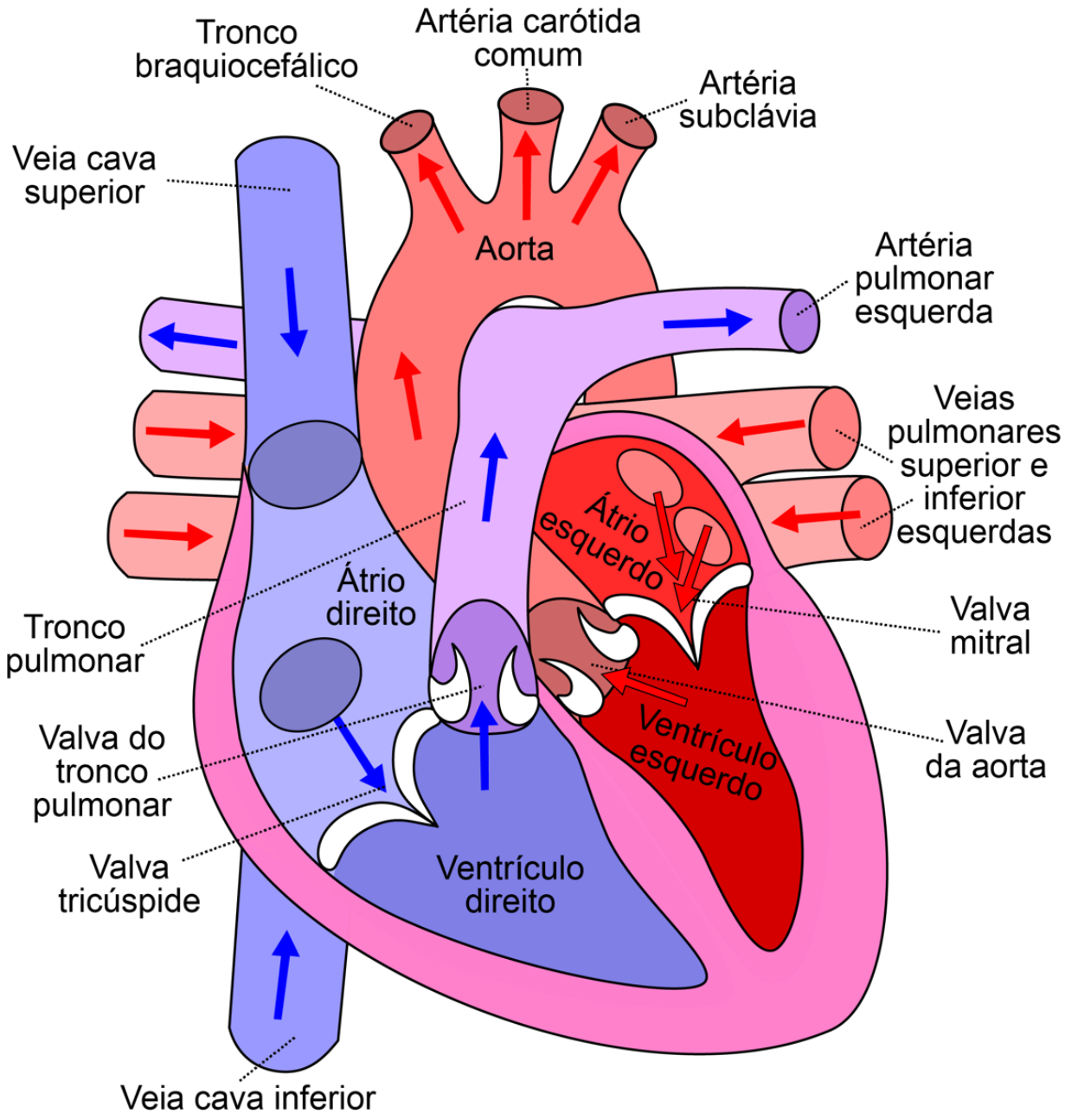


Figura 2.3: Diagrama de um coração humano - fonte: http://bit.ly/diagram-human-heart

Outro bom exemplo é um mapa de transporte urbano:

Figura 2.4: Mapa do metrô de São Paulo - fonte: http://www.metrocptm.com.br/veja-o-mapa-de-estacoes-do-metro-e-cptm/

Em ambos os casos, você está relacionando a posição de alguma coisa com o seu nome. É possível extrair desses exemplos a ordem das estações de metrô, por exemplo, e quais as suas conexões. No diagrama do coração, você conhece a exata posição das partes do coração e suas relações.

Esse modo chega bastante perto do que seria as Triplas da Web Semântica, que veremos mais à frente ainda nesse capítulo.

2.4 Relacionando informações

Há outro passo que precisamos considerar quando organizamos informação. A técnica utilizada para organizar a informação é importante, mas mais importante que isso é **como** você relaciona uma informação com outra. Vamos tentar manter o problema simples, pensando em como seria possível organizar uma informação em formato de texto. Lidamos com textos todos os dias, por isso, não deve ser difícil para você tentar pensar formas de organização.

Existem duas maneiras simples de fazer isso: de forma linear e de forma não linear. A forma linear é como, por exemplo, organizamos um livro. Você inicia a leitura do livro pela primeira página, avançando pelo primeiro capítulo e lê até o último. É muito difícil compreender a história iniciando a leitura pelo meio do livro. Mesmo assim, embora as informações sejam organizadas de forma linear, elas são organizadas (categorizadas) por capítulos e páginas, em que você consegue localizar um pedaço da história quando for necessário.

A Bíblia é um livro que usa diversas técnicas para organizar seu conteúdo. Ela é organizada mais ou menos em ordem de sua história cronológica, e não na ordem em que os livros foram escritos. Digo isso porque há, por exemplo, indícios de que o livro de Jó foi escrito antes do livro de Gênesis.

A Bíblia tem 66 livros, divididos basicamente em 2 partes, sendo o Novo Testamento e o Velho Testamento. No Velho Testamento, existem 39 livros, divididos em 5 categorias: Pentateuco, Livros Históricos, Livros Poéticos, Profetas Maiores e Profetas Menores. A ideia de organizar os textos em capítulos e versículos foi criado pelos tradutores. Já o Novo Testamento tem 27 livros, separados pelas seguintes categorias: Evangelhos, Histórico, Cartas de Paulo, Cartas Gerais e Profético.

A Bíblia é um exemplo bem completo sobre como podemos organizar uma coleção de livros. A organização em capítulos e versículos é útil para encontrar pedaços específicos de textos. Ainda há algumas Bíblias que têm referências no meio do texto, indicando uma relação em outra parte da história ou até em outro livro da Bíblia. Se formos olhar bem a micro-organização dos textos, chegaremos a estudar os quiasmas Bíblicos, mas esse não é o caso agora. Porém, fica a dica para os mais curiosos.

Se basear por referências para organizar informações é a base da estrutura das enciclopédias. As enciclopédias, como a Bíblia, usam mais de uma técnica para organizar a informação: primeiramente as informações estão organizadas em ordem alfabética.

Encyclopædia Britannica;

OR, A

DICTIONARY

OF

ARTS and SCIENCES,

COMPILED UPON A NEW PLAN.

IN WHICH

The different SCIENCES and ARTS are digeſted into diſtinct Treatiſes or Syſtems;

AND

The various TECHNICAL TERMS, *&c.* are explained as they occur in the order of the Alphabet.

ILLUSTRATED WITH ONE HUNDRED AND SIXTY COPPERPLATES.

By a SOCIETY *of* GENTLEMEN *in* SCOTLAND.

IN THREE VOLUMES.

VOL. I.

EDINBURGH:

Printed for A. BELL and C. MACFARQUHAR;
And ſold by COLIN MACFARQUHAR, at his Printing-office, Nicolſon-ſtreet.

M.DCC.LXXI.

Figura 2.5: Primeira edição da Encyclopædia Britannica - https://en.wikipedia.org/wiki/Encyclopædia_Britannica_First_Edition

Era bastante comum haver uma referência a um assunto relacionado àquele que estava sendo lido, ao final do artigo, facilitando a procura por informações de um assunto de mesmo gênero. Quando criança, eu usava muito a enciclopédia Barsa, que só a minha tia tinha, para trabalhos de escola. Sempre juntava um monte de papel almaço (lembra?) e ia para a casa dela tentar encontrar qualquer informação sobre o assunto a ser estudado.

As enciclopédias possuem informações organizadas de forma não linear, ou seja, as informações não estão em uma ordem cronológica, como uma história, mas estão organizadas em ordem alfabética e também de forma relacional e associativa. Exemplo: quando você procura informações sobre "veículos automotores" em uma enciclopédia, ao final do texto, você pode encontrar uma série de referências dos assuntos relacionados, como por exemplo: "motores de combustão interna", "rodas", "tipos de combustíveis", "mecânica" etc.

Essa maneira de associar uma informação com outra é mais ou menos como nosso cérebro funciona. Quando você pensa em um assunto, seu cérebro faz uma série de associações para formar uma ideia, trazer à tona uma memória ou uma lembrança. Por esse motivo, seu cérebro consegue guardar informações que podem ser recuperadas quando pensamos diretamente nelas, ou quando pensamos em assuntos relacionados.

Entendendo que não é mais problema armazenar ou recuperar nossos dados, precisamos pensar em como podemos relacionar e reutilizar esses dados na internet.

2.5 Vannevar Bush e o Memex

Em 1945, foi publicado na revista Atlantic Monthly um artigo chamado **As We May Think** (BUSH, 1945) pelo Doutor e pesquisador Vannevar Bush. No artigo, ele discute um dos problemas mais difíceis da comunidade científica da época: encontrar uma forma de armazenar e recuperar o conhecimento produzido por pesquisas e investigações na área científica.

Na época, os sistemas guardavam as informações de forma hierárquica e muito linear, dificultando pesquisas e a consulta dessas informações. Vannevar Bush dizia que o nosso cérebro guarda e usa a informação de maneira fragmentada, ou seja, o pensamento humano não funciona de maneira linear, mas de forma associativa e relacional (enciclopédias, lembra?).

Quando você conversa com alguém, seu cérebro faz associações buscando informações relacionadas ao assunto. Você traz à tona memórias, lembra de conversas anteriores, assuntos que estudou, informações que recebeu recentemente e assim por diante.

Mas se organizarmos a informação não de forma categorizada ou hierarquizada, mas se baseando por associações e relações, poderíamos obter mais vantagens. Tendo isso em mente, ele propôs uma máquina chamada Memex.

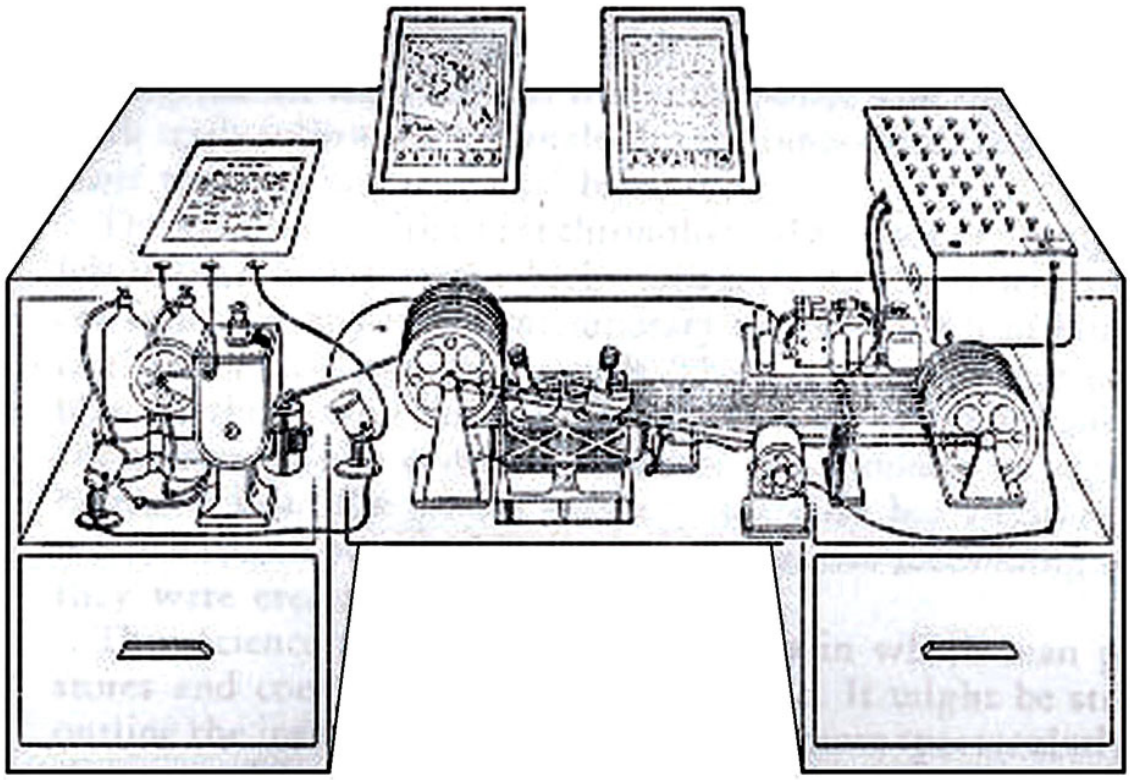


Figura 2.6: Memex- fonte: http://hackeducation.com/2015/05/02/memory-machines

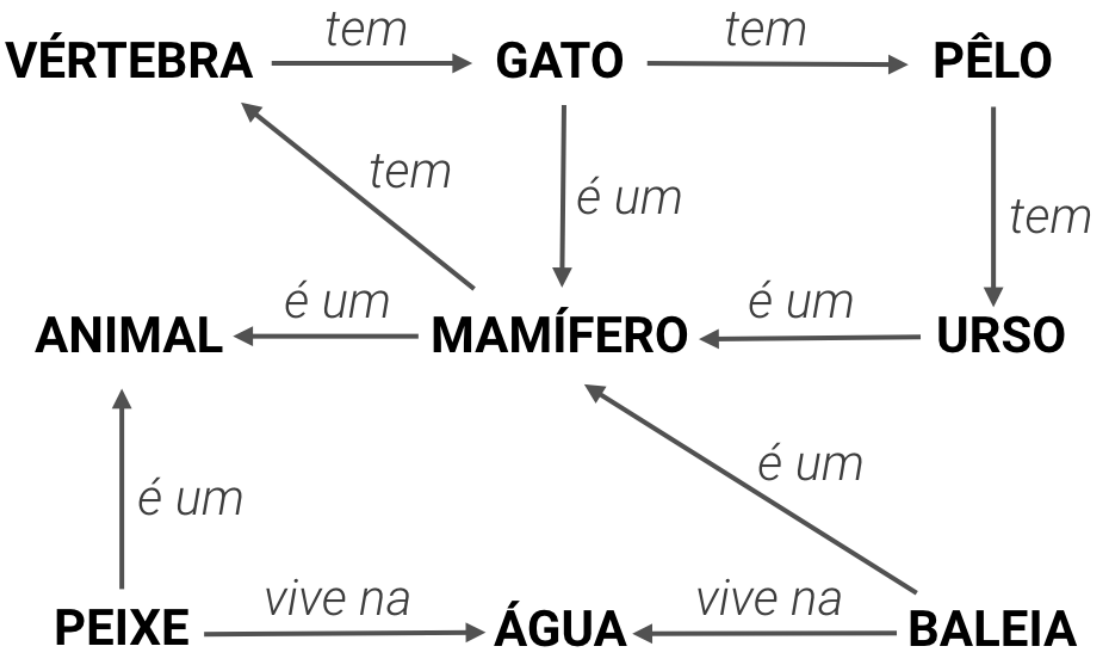
O objetivo de construir o Memex era dar a possibilidade de as pessoas consultarem e relacionarem informações. Por exemplo: você poderia procurar informações sobre veículos automotores, e depois relacionar esse conteúdo com informações sobre combustão interna ou qualquer outro assunto relacionado.

Na verdade, tanto naquele tempo como ainda hoje, a forma com que lidamos com a informação ainda é bem ruim. Nós produzimos muita informação, mas não organizamos de maneira adequada, de forma que possamos relacioná-la e consultá-la. As pessoas não dão a devida atenção para os dados que elas próprias geram todos os dias, e as empresas que têm informações importantes não distribuem essa informação.

O que os sistemas de busca nos dão hoje são apenas resultados, não relevantes, de termos que estão listados em websites. Você

ainda precisa fazer um filtro manual pelos resultados, para encontrar o website que realmente seja relevante. Os buscadores tentam usar algoritmos para dar alguma relevância para os websites selecionados nos resultados, sendo que na verdade esperamos mais relevância no conteúdo encontrado.

Quando relacionamos uma informação com outra, nós aumentamos o contexto de relevância para o termo buscado. O Memex era uma máquina que usava a ideia de relacionamento para consultar e organizar informação. Esse “formato” de relacionar informação para facilitar a sua consulta seria conhecido mais para a frente como **Hipertexto**, termo cunhado pelo Ted Nelson alguns anos depois de Vannevar Bush.



2.6 Douglas Engelbart e Augmenting Human Intellect

Embora o Memex nunca tenha sido construído, um cara chamado **Douglas Engelbart** leu o artigo *As We May Think*, escrito pelo Vannevar Bush, e se inspirou de tal forma que direcionou sua vida para colocar as ideias escritas no artigo em prática. No dia 9 de dezembro de 1968, no Laboratório de Pesquisa de Stanford, ele apresentou para o mundo um sistema chamado NLS (oN-Line System).

Isso é tão importante, que essa apresentação ficou conhecida como **The Mother of All Demos**. Se você assistir ao vídeo (https://www.youtube.com/watch?v=yJDv-zdhzMY), vai entender o motivo.

Nesse dia, Douglas Engelbart apresentou ao mundo o primeiro conceito de estrutura de dados e relacionamento de informação em uma espécie de computador pessoal, com direito a mouse e teclado. Talvez tenha sido a primeira videoconferência, em que ele usou um headset com microfone, além de um processador de texto e uma espécie de editor de código.

Douglas deu o nome desse projeto de **Augmenting Human Intellect** (ENGELBART, 1962). O problema era que, naquela época, para fazer um projeto de estruturação e relacionamento de dados, gastavam-se alguns milhões. A ideia era que a máquina ajudasse a aumentar a capacidade da humanidade de guardar e pesquisar informação de maneira simples. Nessa apresentação, ele começa a organizar as informações em categorias e mostrava como é “fácil” reutilizar e relacionar essas informações, de forma que ele consiga recuperá-las futuramente.

2.7 Ted Nelson, Projeto Xanadu e o início do Hipertexto

Embora Douglas Engelbart tenha levado à pratica um pouco da teoria e dos estudos conhecidos até então, mostrando que toda essa ideia não era algo impossível, ainda era necessário ver algo mais palpável, global, simples de entender. Praticamente na mesma época que Engelbart mostrou para o mundo o “primeiro computador”, Ted Nelson cunhava o termo Hipertexto.

Ted Nelson tinha uma máxima que ele dizia sempre (como se eu conhecesse o cara):

> *Most people are fools, most authority is malignant, God does not exist, and everything is wrong.*

Sim, ele era polêmico e também era conhecido por iniciar vários projetos e nunca terminar nenhum. Faça uma pesquisa sobre ele e leia alguns de seus comentários e artigos, você vai perceber que ele é um cara bem ácido às vezes. Veja esse comentário raivoso (http://hyperland.com/TBLpage) sobre uma citação bem respeitosa que o próprio Tim Berners-Lee fez sobre ele:

> *No one has ever paid me to be a visionary… It is vital to point out that Tim's view of hypertext (only one-way links, invisible and not allowed to overlap) is entirely different from mine (visible, unbreaking n-way links by any parties, all content legally reweavable by anyone into new documents with paths back to the originals, and transclusions as well as links-- as in Vannevar Bush's original vision).* - http://hyperland.com/TBLpage

Em 1960, ele criou um projeto chamado Xanadu (http://www.xanadu.ne). Este tinha como proposta construir um sistema que o mundo pudesse usar para organizar e gerenciar

dados. O ponto é que não era apenas simplificar a conexão das ideias, mas representar mais claramente e corretamente a informação, tentando substituir não apenas o papel como mídia, por uma família totalmente diferente de estruturação de informação.

O pessoal envolvido no projeto, principalmente o Ted Nelson, falava que a ideia não era fazer um World Wide Web, mas sim prevenir a criação de uma!

> *The World Wide Web was not what we were working toward, it was what we were trying to* ***PREVENT****. The Web displaced our principled model with something far more raw, chaotic and short-sighted. Its one-way breaking links glorified and fetishized as "websites" those very hierarchical directories from which we sought to free users, and discarded the ideas of stable publishing, annotation, two-way connection and trackable change.*
>
> Fonte: http://www.xanadu.com.au/ted/XUsurvey/xuDation.html

Xanadu consistia em uma espécie de rede mundial que permitiria que a informação fosse guardada não como arquivos separados, mas como documentos conectados. Quando você fosse escrever algum texto, por exemplo, você poderia relacionar um pedaço desse texto com outro arquivo, que poderia estar em outro computador em algum lugar do mundo. Nos casos onde as informações tivessem diretos de copyright, os autores originais seriam pagos automaticamente pela cópia virtual de seus documentos. Eu não sei bem como ia funcionar, fica até uma tarefa para se aprofundar mais no assunto e tentar descobrir, mas seria uma espécie de pay-per-view de conteúdo.

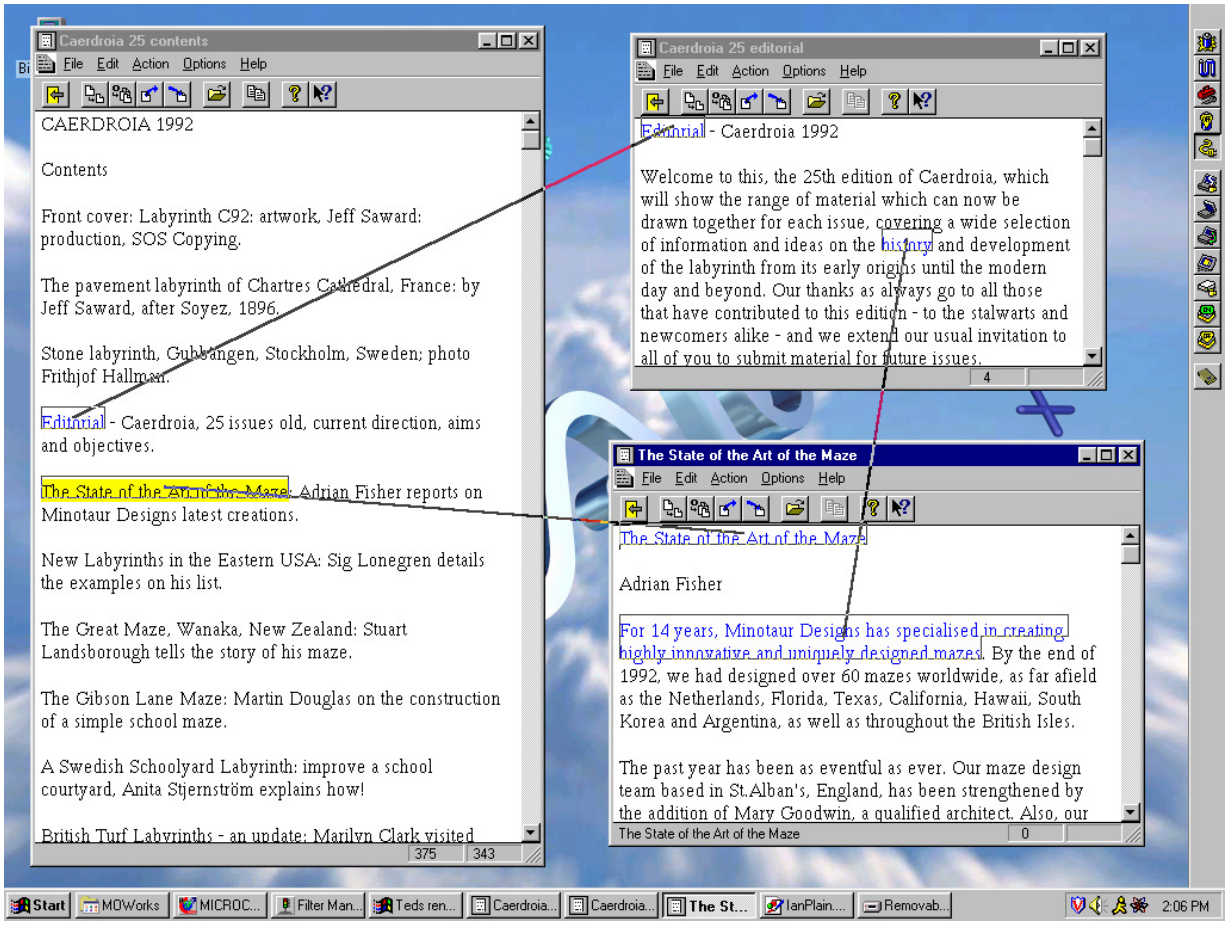

Caerdroia 25 contents
File Edit Action Options Help
CAERDROIA 1992
Contents
Front cover: Labyrinth C92: artwork, Jeff Saward: production, SOS Copying.
The pavement labyrinth of Chartres Cathedral, France: by Jeff Saward, after Soyez, 1896.
Stone labyrinth, Gubbängen, Stockholm, Sweden; photo Frithjof Hallman.
Editorial - Caerdroia, 25 issues old, current direction, aims and objectives.
The State of the Art of the Maze: Adrian Fisher reports on Minotaur Designs latest creations.
New Labyrinths in the Eastern USA: Sig Lonegren details the examples on his list.
The Great Maze, Wanaka, New Zealand: Stuart Landsborough tells the story of his maze.
The Gibson Lane Maze: Martin Douglas on the construction of a simple school maze.
A Swedish Schoolyard Labyrinth: improve a school courtyard, Anita Stjernström explains how!
British Turf Labyrinths - an update: Marilyn Clark visited
375 343
Caerdroia 25 editorial
File Edit Action Options Help
Editorial - Caerdroia 1992
Welcome to this, the 25th edition of Caerdroia, which will show the range of material which can now be drawn together for each issue, covering a wide selection of information and ideas on the history and development of the labyrinth from its early origins until the modern day and beyond. Our thanks as always go to all those that have contributed to this edition - to the stalwarts and newcomers alike - and we extend our usual invitation to all of you to submit material for future issues.
4
The State of the Art of the Maze
File Edit Action Options Help
The State of the Art of the Maze
Adrian Fisher
For 14 years, Minotaur Designs has specialised in creating highly innovative and uniquely designed mazes. By the end of 1992, we had designed over 60 mazes worldwide, as far afield as the Netherlands, Florida, Texas, California, Hawaii, South Korea and Argentina, as well as throughout the British Isles.
The past year has been as eventful as ever. Our maze design team based in St.Alban's, England, has been strengthened by the addition of Mary Goodwin, a qualified architect. Also, our
The State of the Art of the Maze
0
Start
MOWorks
MICROC...
Filter Man...
Teds ren...
Caerdroia...
Caerdroia...
The St...
IanPlain...
Removab...
2:06 PM

Figura 2.8: Fonte: http://www.xanadu.com.au/ted/XUsurvey/xuDation.html

O screenshot anterior é de uma demo de 1998, do conceito chamado **Transpointing Windows**, em que o leitor veria conexões visíveis entre o conteúdo de um documento e outro. O projeto tinha uma estrutura de Hipertexto que eles planejaram, onde existiriam links que nunca se quebrariam, mas poderiam mudar de versão, na qual documentos poderiam ser comparados lado a lado. Inclusive, estes poderiam conter anotações dos leitores; além de ser possível ver a origem de cada uma das citações.

Por causa de problemas financeiros e várias confusões internas, o projeto não foi lançado e perdeu a oportunidade de ter sido o sistema padrão mundial de Hipertexto. O HTML ganhou e Ted Nelson não gostou muito. Ele diz que o HTML é um sistema ruim, que trata a ideia inicial (que é relacionar informação) do Hipertexto

de forma trivial, com links frágeis, que quebram facilmente levando a lugar algum, que não reconhecem as mudanças das fontes originais e nem tem um sistema inteligente de pagamento de direitos de copyrights, nem princípios de multiversionamento do conteúdo ou sua reutilização.

Devo confessar que, até hoje, eu não consegui entender bem o projeto de Ted Nelson. Em muitas formas, ele parece ser mais complicado do que toda a ideia do HTML e da internet como ela foi concebida. Talvez, a única coisa que sobrou de bom nessa história toda foi o conceito de interconexões de informações de diversas fontes.

O Projeto Xanadu já não existe mais. Você pode ver alguns detalhes de como tudo funcionava ainda no site deles (http://xanadu.com.au/ted/XUsurvey/xuDation.html). Mas a ideia de Hipertexto, na qual várias fontes de informação são relacionadas, ainda persiste.

Embora não seja a ideia original do Ted Nelson, acho que o HTML e a Web, como são hoje, fazem um ótimo trabalho. Talvez o Projeto Xanadu nos levaria para uma internet com mais controle das empresas, menos democrática e mais inflexível. Contudo, não dá para negar que optar por um sistema mais flexível, anônimo e democrático traga uma série de problemas, como esse que estamos discutindo aqui: um padrão de organização da informação e seus relacionamentos em nível global.

A Web Semântica é uma proposta para organizar a informação da Web toda? Isso não é meio utópico? O que realmente é a Web Semântica, então? É o que vamos ver no próximo capítulo.

2.8 Para ler mais

- Timeline do Hipertexto — https://en.wikipedia.org/wiki/Timeline_of_hypertext_technology
- Artigo sobre o Cérebro Humano na Super Interessante — http://super.abril.com.br/ciencia/cerebro-humano
- Know your brain — http://www.ninds.nih.gov/disorders/brain_basics/know_your_brain.htm
- Apresentação do Artigo As We May Think - Vannevar Bush — http://pt.slideshare.net/dmartins/apresentao-do-artigo-as-we-may-think-vannevar-bush
- The Web Before the Web — http://1997.webhistory.org/historyday/abstracts.html#doug
- Vídeo - Animação do que seria o Memex — https://www.youtube.com/watch?v=c539cK58ees

- BUSH, Vannevar. *As we may think.* 1945. Disponível em: http://www.uff.br/ppgci/editais/bushmaythink.pdf.

- ENGELBART, Doug. *Augmenting Human Intellect*. 1962. Disponível em: http://www.dougengelbart.org/pubs/augment-3906.html.

- RITCHIE, Ian. *The day I turned down Tim Berners-Lee.* 2011. Disponível em: https://www.ted.com/talks/ian_ritchie_the_day_i_turned_down_tim_berners_lee/.

- ROSLING, Hans. Hans Rosling shows the best stats you've ever seen. 2006. Disponível em: http://bit.ly/hansrosling-ted.

- SIEGLER, M. G. *Eric Schmidt:* Every 2 Days We Create As Much Information As We Did Up To 2003. 2010. Disponível em: https://techcrunch.com/2010/08/04/schmidt-data

- SINTEF. *Big Data, for better or worse:* 90% of world's data generated over last two years. ScienceDaily, 22 May 2013. Disponível em: www.sciencedaily.com/releases/2013/05/130522085217.htm.

Capítulo 3
O que é Web Semântica?

> *I think what's really amazing is that given the scale of the web and getting the computer power we have today, we're starting to see things that appear intelligent but actually aren't semantically intelligent.* — **Marissa Mayer**

3.1 Qual o problema da Web atual?

Por que precisamos da Web Semântica? A Web de hoje não é suficiente? Essas são perguntas que qualquer um se faz quando começa seus estudos sobre Web Semântica.

Primeiro, quero pedir para que você não confunda código semântico com Web Semântica. Código semântico é o ato de escrever um código de marcação corretamente, utilizando as tags corretas para suas determinadas funções, isso é: usar tags de título para marcar informações que sejam títulos e assim por diante. Web Semântica depende de código semântico? Não. Mas ajuda bastante.

A Web Semântica **não depende** do código HTML bem escrito para existir. Todos os desenvolvedores precisariam ser rígidos com seus códigos, usando perfeitamente todas as tags do HTML em suas respectivas funções. E isso é impossível.

Eric Miller e Ralph Swick descrevem a Web Semântica como *“uma extensão para a web atual, na qual o significado da informa ão é clara e explicitamente ligada a partir dela mesma, permitindo que computadores e pessoas trabalhem em coopera ão”.* (http://onlinelibrary.wiley.com/doi/10.1002/bult.280/full) Logo,

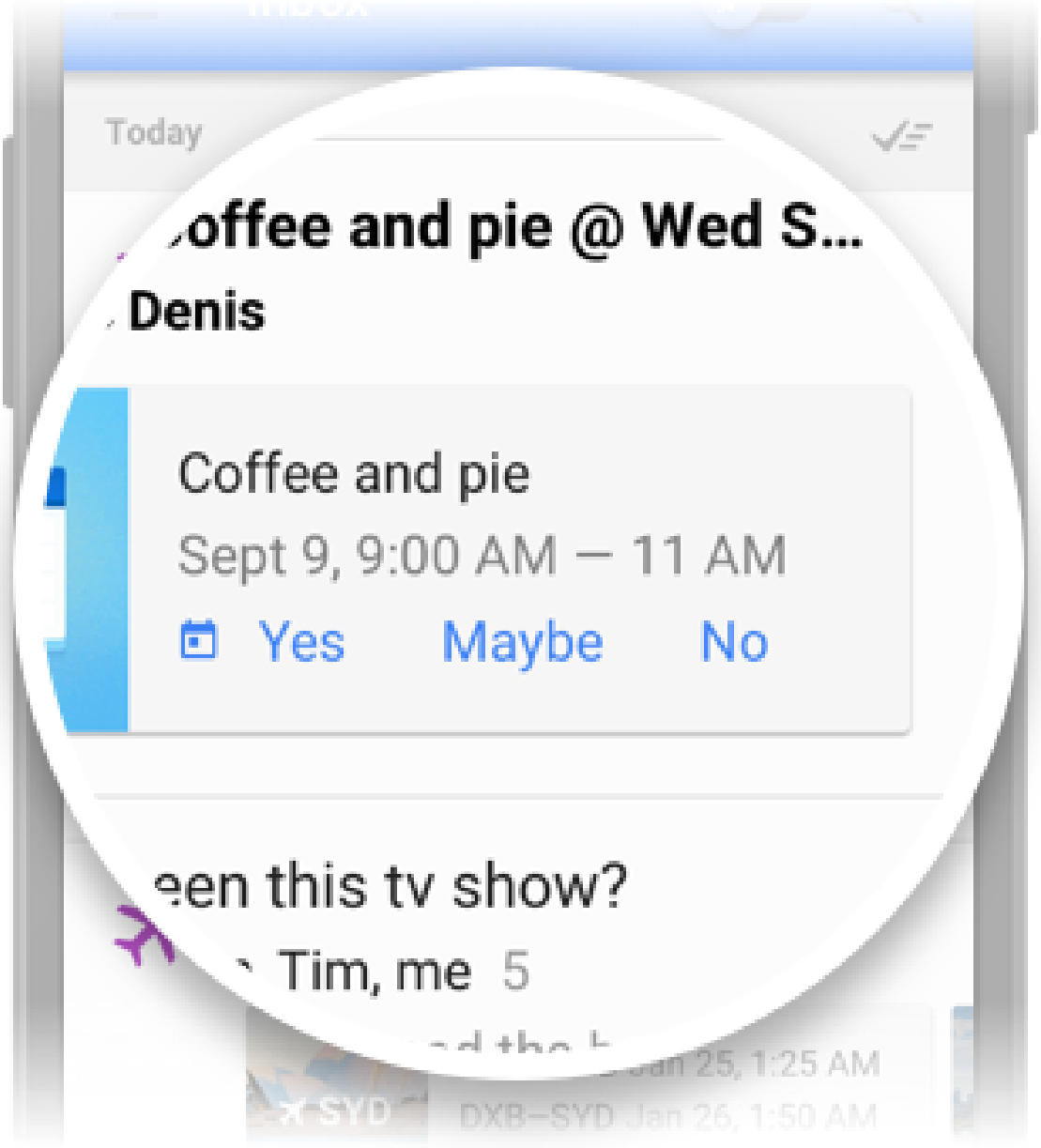
entenda que o conceito da Web Semântica transcende o código semântico.

A ideia de que seu código HTML deve ser semântico para que sistemas e robôs possam extrair mais facilmente informação das suas páginas ainda é verdadeira, mas nós não usamos especificamente suas tags para organizar a informação na Web. Mesmo porque formatos de estruturas de dados como JSON-LD (JSON for Linked Data - que veremos em capítulos posteriores) podem ser usados em qualquer código de marcação.

Realmente nós não sabemos de verdade qual o impacto que a Web Semântica terá nas nossas vidas, exatamente porque ela ainda não existe de forma plena. Mas nós temos algumas pistas do que seria Web Semântica todas as vezes que usamos serviços como Gmail. Se você já experimentou usar o Google Now, já deve ter levado alguns sustos. Eu, pelo menos, levei quando ele me avisou que meu voo ia atrasar, pelo simples fato de ele ter interpretado o e-mail que continha informações da minha passagem aérea.

Existem alguns e-mails que são mostrados de maneira diferente, por exemplo, e-mails com cartões de embarque, ingressos, entradas para cinema e até alguns de listas de discussão. Todos esses e-mails são customizáveis por quem os envia. Essa customização é feita enviando um JSON-LD (que vamos ver mais à frente no livro). O Google interpreta esse JSON com as suas informações e formata o e-mail de acordo.

Se você envia para alguém um convite para um evento, reunião etc., o e-mail da pessoa apareceria assim:



A alma da Web Semântica é colocar as máquinas para trabalhar para você. É melhorar a vida dos usuários e facilitar o máximo a execução de nossas tarefas. Hoje, isso é impossível. A Web que frequentamos hoje é muito passiva, embora nós possamos interagir muito mais do que anos atrás.

Para entender as fases que a Web já passou, os estudiosos separam-na em pelo menos 3 fases: Web 1.0 (passiva), Web 2.0 (a nossa atual, problemática), Web 3.0 (Web Semântica).

3.2 A Web 1.0 — Web passiva

A Web 1.0 foi a primeira fase da Web. É uma Web em que a única função do usuário era consumir. A ideia dessa primeira fase foi realmente organizar a informação de forma que ela ficasse disponível para o consumo de qualquer usuário que estivesse conectado na internet.

Não tem muito segredo nessa fase. Você entra na internet e consome seu conteúdo. Os motores de busca ficaram muito populares aqui, já que foram usados como ponto de partida para navegar na web em muitos momentos. Mesmo assim, a internet ainda era uma plataforma de mão única, na qual os usuários apenas consumiam o que era criado e produzido por outros.

Hoje, percebemos que essa primeira fase foi um problema exatamente por ela ter sido unilateral, ou seja, uma parte comanda o que será consumido pela outra parte. Mas, ao meu ver, as pessoas não estavam preparadas para produzir conteúdo como estavam na Web 2.0. Acho que o ciclo natural das coisas fez bem para a evolução dos usuários, criando um senso crítico que seria muito praticado na segunda fase que viria em breve.

3.3 A Web 2.0 — Problema atual da Web

Pense duas vezes se você acha que toda aquela história de Web 2.0 é um blá blá blá gigante. Embora o termo tenha sido muito banalizado, ele define uma fase importante da história da Web.

Essa fase mudou a internet da água para o vinho. A Web se tornou uma plataforma de compartilhamento e criação de conteúdo para os usuários. Nesse momento, as pessoas passaram de consumidoras para produtoras de conteúdo. Desde um simples blog que alguém comum usa para publicar seu conteúdo, permitindo que pessoas

normais comentem sobre o assunto, até as grandes redes sociais, que criaram uma plataforma completa para que seus usuários pudessem produzir conteúdo e expor suas opiniões publicamente.

Ainda me lembro muito do início dos blogs aqui no Brasil. Foi emocionante ver como pessoas comuns, com opiniões inteligentes, começaram a ficar bem populares no mercado brasileiro. Foi uma série de pessoas que aproveitou a onda da Web 2.0 para expor sua opinião, e principalmente estabelecer um meio de comunicação com seus leitores.

Na minha opinião, o blog foi o responsável por preparar um campo muito próspero de produção de conteúdo independente, coisa que plataformas como o YouTube nunca será, embora seja muito mais popular hoje em dia.

Contudo, são públicos bem diferentes para épocas diferentes. O YouTube foi fundado em 2005, ainda muito tímido, foi comprado pelo Google em 2006. Nessa época, os blogs já estavam no auge. O termo blog não foi cunhado antes de 1990. Achei no blog do Cris Dias, famosíssimo, muito gente boa, um post do ano 2000 (DIAS, 2000). Eu não sei se ele começou antes, o que é muito provável, mas os blogs já estavam se popularizando bem nessa época.

Jornalistas tinham um medo terrível na época de que o Blog iria matar o jornalismo, já que qualquer um podia expor sua opinião na internet. Como os blogs estavam populares, as pessoas preferiam ouvir as opiniões de pessoas próximas do que dos grandes meios de comunicação.

A Web 2.0 focou na colaboração e compartilhamento de informação entre pessoas comuns, sem depender dos grandes meios de comunicação para tal. Foi, e ainda é, uma Web interativa, dinâmica, em que as indústrias têm pouca voz e, por isso, tentam se misturar nas redes sociais, presenças online etc.

O termo Web 2.0 foi cunhado pela empresa americana O’Reilly Media (O'REILLY, 2005). As duas principais características são:

- Grande parte do conteúdo é criado, produzido e compartilhado pelos usuários.
- A Web se transforma em uma plataforma para compartilhamento de conteúdo: imagem, voz, texto, vídeo.

Outras caraterísticas também foram citadas nessa fase, como o *The Perpetual Beta* ou o *Always beta*. Quem usou o Gmail desde o início sabe que ele fixou a palavra **beta** logo abaixo do seu logo durante muito tempo. Esse conceito diz que como os dispositivos e principalmente softwares estão ligados à internet, sendo oferecidos como serviço, eles não são mais um software individual, de caixa, oferecido por versões. Eles passam a evoluir constantemente, engajando usuários como seus principais testadores e subindo frequentemente novas features para o serviço.

O pessoal do Agile fala muito disso hoje sob o nome de **entregas incrementais**. Podemos também citar o **continuous delivery**, já que a ideia é entregar pequenas partes, incrementais, que podem ser entregues ao cliente mais frequentemente.

> **ENTREGA INCREMENTAL** não é a mesma coisa que entrega contínua. Quando falamos sobre entregas incrementais, estamos nos referindo ao desenvolvimento de partes pequenas e funcionais de um projeto. Essas partes pequenas podem ser entregues hoje ou daqui meses. Aí vem a **entrega contínua**, que prega a entrega dessas partes de forma frequente e previsível.

Outra coisa importante é o conceito de **cooperar, não controlar**. Podemos dizer que foi o *start* para que qualquer serviço de internet tivesse uma API de forma que outros serviços pudessem utilizar seus dados. Amazon, Google e outros começaram a oferecer maneiras de que tanto o público quanto outras empresas pudessem plugar suas tecnologias. Daí vemos empresas como a Amazon provendo suas informações via SOAP, ou até um serviço REST.

Muitas outras tecnologias emergiram nessa época, como o Ruby e o Rails, por exemplo. A leitura de Feeds RSS (e depois Atom) ficaram populares nessa fase dos blogs. Se você não sente saudades do Bloglines, é por que perdeu um tempo feliz na internet.

Só que há um problema nisso tudo: na web atual que, digamos, seria o final da web 2.0, os dados que todo mundo consegue consumir na distância de um click são apenas compreendidos e reutilizados por seres humanos. As máquinas (e quando digo máquinas, quero dizer robôs, sistemas, programas, computadores etc.) não conseguem entender toda essa informação disponível.

Eles não têm contexto, e o máximo que se pode falar é que as máquinas têm pistas sobre a informação encontrada na internet. O Google até consegue dizer que um parágrafo não é um título, mas não consegue dizer sobre qual assunto esse texto está se referindo.

É por isso que vamos para uma próxima fase, que os estudiosos de plantão sugerem o nome de Web 3.0. Esta é uma web na qual toda a informação pública e disponível na internet deve ser entendida pelas máquinas.

Você já deve ter ouvido falar sobre Mashups. O nome já não é tão usado assim hoje em dia, mas lá por volta de 2006 a 2008 (GREENE, 2008), as Mashups apareceram e morreram. Mashups são serviços que usam conteúdos de vários outros serviços, por meio de suas APIs abertas, para gerar mais informação. Vão desde serviços simples, como agregadores de conteúdo, tipo o Flipboard, até serviços que cruzavam informações de ocorrências policiais da cidade, mostrando onde elas aconteceram direto em mapas do Google Maps. Existiam vários serviços como o Yahoo Pipes, MashMaker e outros que possibilitavam a combinação de informações e funções de vários serviços.

Hoje, o que chega mais perto dessa ideia são serviços como o IFTTT (*If This Than That*) (https://ifttt.com), Zapier (https://zapier.com) e Piesync (http://www.piesync.com), que usam

as APIs abertas de diversos sistemas online, para que eles trabalhem para você. Eles possibilitam que você faça coisas simples, como por exemplo, sincronizar suas fotos em redes sociais até controlar as luzes e a TV quando você chega na sua casa. A ideia de colocar esses “robôs” para trabalhar sem sua intervenção é só uma pequena amostra do que seria a Web Semântica de verdade, usando informações abertas de toda a internet.

Depois disso, a Web começou a ser tratada como uma verdadeira plataforma, na qual os sistemas não teriam uma versão específica, pois estariam sempre evoluindo e ainda trocando informações entre si.

3.4 A Web 3.0, o início da Web Semântica

Experimente procurar no Google a palavra *Manga*. Veja quantas repostas aparecem. O Google não sabe se esse termo se refere à fruta, a uma parte da sua roupa ou ao mangá japonês. Aqui aparecem uma série de resultados misturados, sobre os três assuntos.

Isso acontece porque a máquina não tem o contexto da sua busca. Com a vinda do HTML 5 e suas novas tags, conseguimos levar parte do significado da informação para a estrutura da página. Logo, os buscadores, que se beneficiam imediatamente com essa mudança, conseguem entender que um determinado parágrafo faz parte do artigo da página, mas ele não sabe ainda se esse artigo fala sobre um filme, uma pessoa, um lugar, um objeto etc.

A Web 3.0 é o que chamaríamos de Web Semântica. Muitos estudiosos falam que ela é uma espécie de extensão da nossa web atual, em que os usuários não apenas lerão e produzirão conteúdo, mas usarão as máquinas para relacionar e reutilizar esse conteúdo da melhor maneira possível. É uma combinação da Web 2.0,

criando uma nova fase na qual as máquinas usarão toda a informação que está disponível na internet.

Hoje, a maior parte da informação guardada na internet está em uma linguagem natural e grande parte da informação útil, que poderia ser usada pelas máquinas, está presa em bancos de dados dentro das próprias empresas e serviços de internet. A informação disponível está apenas legível para seres humanos.

Você consegue entender um texto na internet porque está escrito em um idioma que você conhece. Desse texto, você consegue saber o que é um título e o que é um parágrafo porque você é um ser humano e sabe diferenciar esses dois tipos de informação. Mas para um computador, ou para um sistema de busca como o Google, um título e um parágrafo são apenas dados. Ele não sabe dizer o que é um e outro. Para ele, são apenas caracteres de uma tabela usada para mostrar letras na tela.

A Web Semântica é baseada na ideia de que o conteúdo deve ter uma descrição digital, padronizada por vocabulários e que provê meios para as máquinas (robôs, sistemas etc.) entenderem do que se trata esse conteúdo, ou seja, criando algum contexto para que as máquinas consigam utilizar esses dados. Dessa forma, os computadores poderão interpretar as informações, gerando e distribuindo conteúdo útil, de acordo com as necessidades dos usuários.

Você vai poder dizer que determinado texto está falando sobre uma pessoa ou uma coisa. O Google vai saber que existem pelo menos três contextos diferentes para o termo **manga**, e vai conseguir separar esses contextos, por exemplo.

Até agora, as empresas viam a Web Semântica como algo utópico, que mora somente no mundo das ideias e que talvez nunca vire realidade de verdade. Mas com a popularização dos *Wearables* e da *Internet das Coisas*, esse cenário utópico tem virado cada vez mais realidade. No próximo capítulo, vamos conhecer um pouco sobre

dados conectados, conhecido como *Linked Data*, conceito base para o JSON-LD (*JSON for Linked Data*).

3.5 Para ler mais

- Vídeo sobre a história da Web Semântica — https://www.youtube.com/watch?v=bd8zR0v7Jts
- Vídeo sobre Machine Learning — https://www.youtube.com/watch?v=gNkE3tFUFuw&list=PL0Smm0jPm9WcCsYvbhPCdizqNKps69W4Z
- Vídeo sobre The Semantic Web: The Inside Story — https://www.youtube.com/watch?v=3Ap5FsxvjTQ
- Vídeo do Workshop de Semântica no Globoesporte — https://www.youtube.com/watch?v=Q1uB66oSMv8
- Vídeo do TEDx, *How Web 3.0 will change our lives?* — https://www.youtube.com/watch?v=u2w9XKASbPw

- ALSHARIF, Muhammad. *Semantic Web Core Technologies.* 2013. Disponível em: http://www.cs.wustl.edu/~jain/cse570-13/ftp/semantic/index.html.

- DIAS, Cristiano. 2000. *Life, death and Everquest…* Disponível em: http://www.crisdias.com/2000/11/21/salon-com-life-death-and-everquest/.

- DISCOVERY, Wod. *Data and the Web – a great many of choices.* 2010. Disponível em: https://webofdata.wordpress.com/2010/03/01/data-and-the-web-choices/.

- GREENE, Kate. *Web Mashups Made Easy.* March 2008. https://www.technologyreview.com/s/409732/web-mashups-made-easy/.

- OLIVEIRA, Renan. *Introdu ão a Web Semântica e o case da Globo.com*. 2012. Disponível em: http://www.slideshare.net/renangpa/introduo-a-web-semntica-e-o-case-da-globocom?ref=https://renanoliveira.net/tag/sparql/.
- O'REILLY, Tim. *Design Patterns and Business Models for the Next Generation of Software.* 2005. Disponível em: http://www.oreilly.com/pub/a/web2/archive/what-is-web-20.html.
- PETKOVA, Teodora. *A Web of People and Machines:* W3C Semantic Web Standards. 2016. Disponível em: http://ontotext.com/a-web-of-people-and-machines-w3c-semantic-web-standards/.
- SHVAIKO, Pavel; GIUNCHIGLIA, Fausto; SILVA, Paulo Pinheiro da; MCGUINNESS, Deborah L. *Web Explanations for Semantic Heterogeneity Discovery.* 2005. Disponível em: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.4.969&rep=rep1&type=pdf
- SIGNER, Beat. *Semantic Web and Web 3.0.* 2016. Disponível em: http://pt.slideshare.net/signer/semantic-web-and-web-30-lecture-10-web-technologies-1019888bnr.
- SPORNY, Manu. *JSON-LD and Why I Hate the Semantic Web.* 2014. Disponível em: http://manu.sporny.org/2014/json-ld-origins-2/.
- THOMAS, Christopher; SHETH, Amit. *On the Expressiveness of the Languages for the Semantic Web* - Making a Case for 'A Little More'. Capturing Intelligence, v. 1, 2006, p. 3-20. Disponível em: http://knoesis.org/students/topher/publications/FLSW01.pdf.

Capítulo 4
Linked Data: o conceito de dados conectados

> *That's just the web, right? I mean, we've had the* `<a href>` *tag since literally the beginning of HTML / The Web. It's for linking documents. Documents are a representation of data.* **— Manu Sporny**

Eu sei que parece um sonho, mas imagine se tivéssemos um grande banco com todos os dados já produzidos disponíveis na internet. Imagine agora que esses dados estão todos conectados de alguma forma e podem ser acessados por qualquer dispositivo ou ser humano. Esse banco de dados global é a meta da Web Semântica. Para que isso seja possível, é necessário que existam padrões para a relação desses dados. Discutiremos isso nos capítulos posteriores, nos quais falaremos sobre RDF e suas serializações, vocabulários etc. Por agora, vamos entender a base dos conceitos de Linked Data, que vai nos ajudar a ter uma visão melhor sobre a Web Semântica e suas tecnologias.

O termo Linked Data refere-se a um pacote de boas práticas para publicar e conectar dados na web. Para se ter uma ideia do status atual da execução desse conceito, existe um termo chamado de **LOD Cloud** — *Linked Open Data Cloud* (http://lod-cloud.net). Esse diagrama mostra o status da conexão de dados estruturados no mundo em 2007:

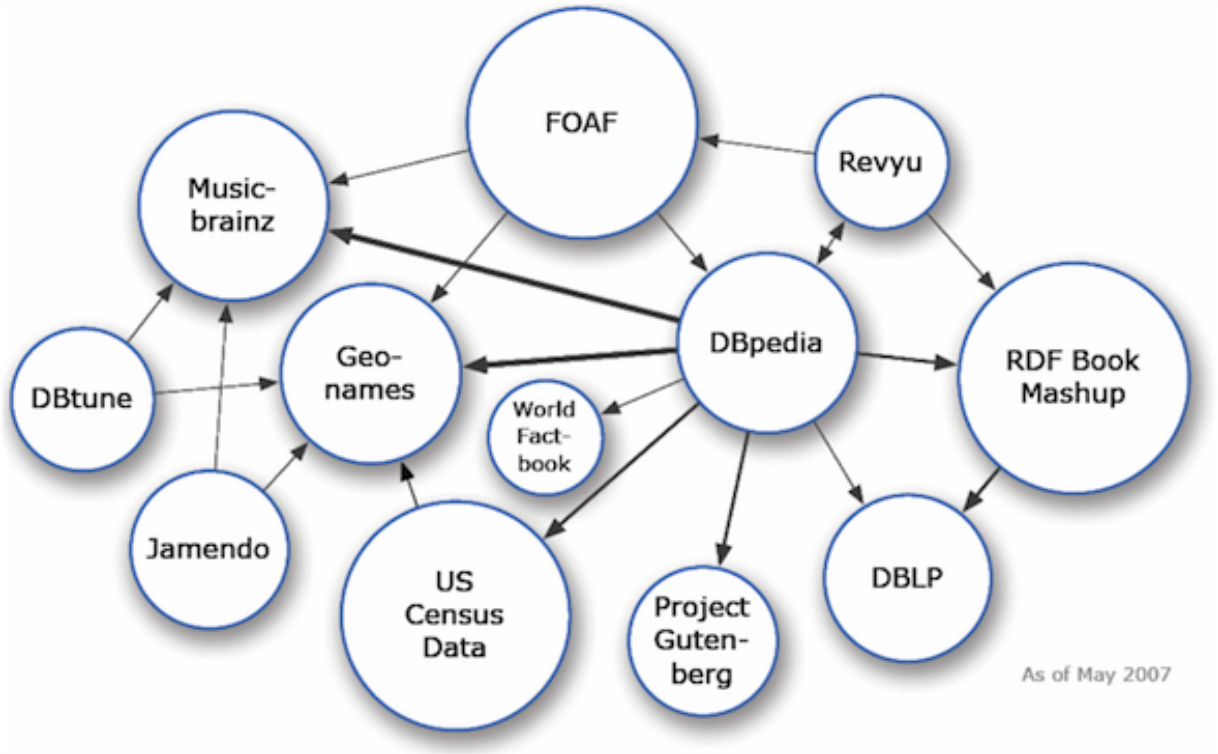


Figura 4.1: Fonte: http://lod-cloud.net/versions/2007-05-01/lod-cloud.png

E este é o status em 2014:

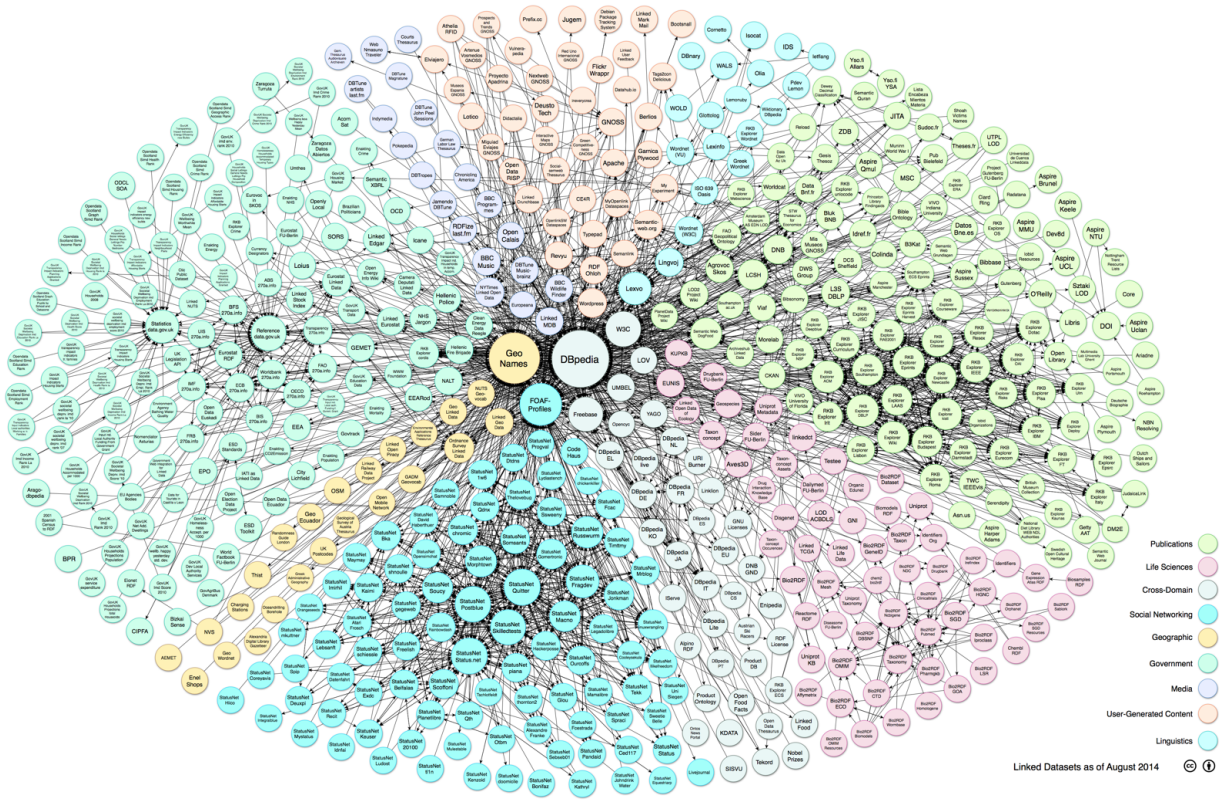


Figura 4.2: Fonte: http://lod-cloud.net/versions/2014-08-30/lod-cloud_colored.png

Perceba que uma movimentação crescente para agrupar, organizar e servir os dados existentes na internet, em um formato padrão, cresceu muito durante 7 anos. Todos esses nós representam um conjunto de dados abertos, os famosos *datasets*, que vamos ver com um pouco mais de detalhes mais à frente no capítulo *Vocabulários*.

Todo esse conceito é suportado por duas tecnologias básicas: a URI para identificar recursos, e o HTTP para descrever esses recursos de informação. Acima dessas duas tecnologias, outras fazem a parte de conexão e estruturação desses dados. O HTML provê uma estrutura e link para documentos na Web; o RDF provê um framework para modelar dados de forma genérica, relacionando por meio das triplas coisas como lugares, pessoas, objetos etc. Não se preocupe, veremos tudo isso durante o livro, o importante agora é entender o conceito de como conectamos os dados na Web Semântica.

4.1 As 4 regras

O conceito de Linked Data é baseado em 4 regras. Essas regras foram escritas em 2006 pelo Tim Berners-Lee, e não devem ficar caducadas tão cedo, já que elas estão intimamente ligadas com a base e os fundamentos da internet. São elas:

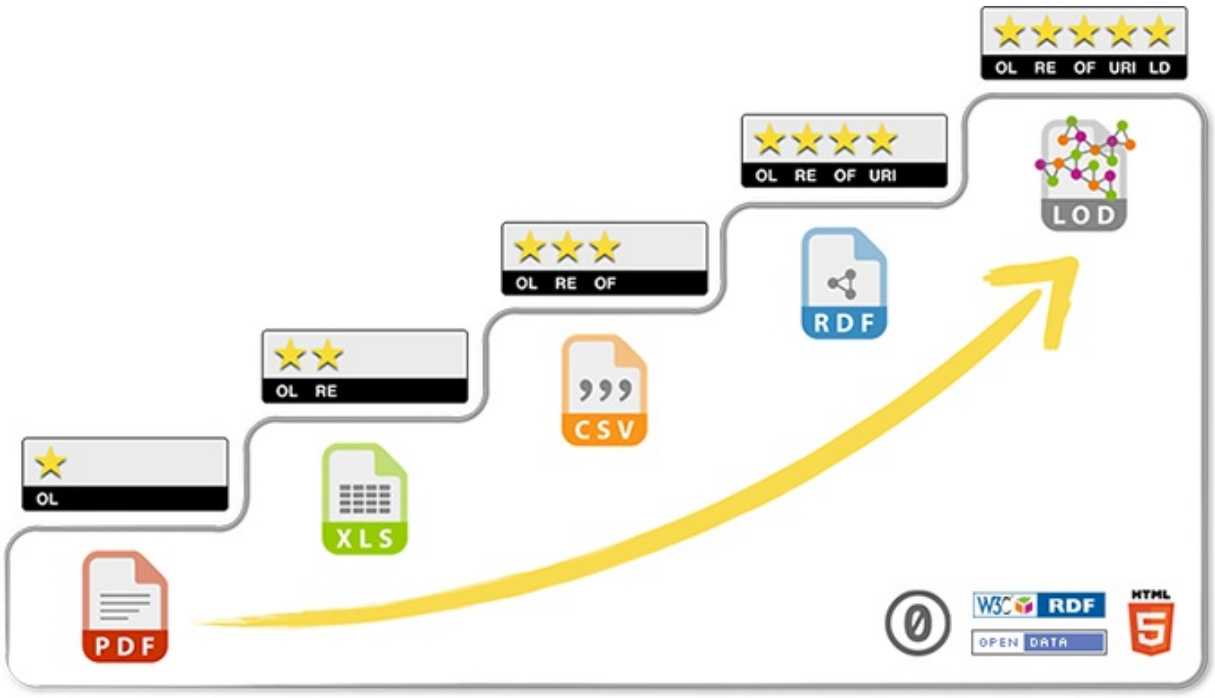
1. Use URIs para nomear as coisas;
2. Use HTTP URIs para que pessoas possam procurar essas coisas;
3. Quando alguém procurar uma URI, disponibilize informação útil, usando padrões como RDF;
4. Inclua links para outras URIs de forma que exija relacionamento entre as fontes de informação;

Simples: identifique, dê um endereço de localização, disponibilize informação e conecte usando links. Já é mais ou menos como a internet funciona hoje e como a Web é baseada. Nós sempre localizamos recursos usando URIs (mais especificamente URLs) e conectamos com links, entretanto, **não** disponibilizamos a informação que temos nos sites de forma que seja reutilizável por outras máquinas e pessoas. Na verdade, os dados ficam visíveis apenas para seres humanos e, mesmo assim, para aqueles que não têm nenhuma deficiência física. Ainda há poucos websites que trabalham com acessibilidade para facilitar o acesso ao conteúdo para pessoas cegas ou com algum tipo de deficiência física.

4.2 Status 5 estrelas

Quebrar uma das quatro regras não destrói a web, mas nós perdemos a oportunidade de fazer com que os dados sejam relacionados. Tim Berners-Lee fez um pequeno questionário de 5 perguntas para medir o quanto seus dados estão abertos. Esse questionário se chama *“Seus Dados são 5 estrelas?”*

(http://5stardata.info/pt-BR/). A cada pergunta positiva, você ganha uma estrela. Se completar 5, você é um cara sensacional. Caso contrário, não se preocupe; procure o que pode ser melhorado e corrija o problema.



Estrela 1

Seus dados estão disponíveis na Web, em qualquer formato, sob uma licença aberta. Você pode disponibilizar um PDF, mas o conteúdo desse PDF não deve estar sob licença de *copyright*, por exemplo. Ele precisa estar aberto, para que a Web possa reutilizar seu conteúdo de diversas formas. Essa primeira estrela é simples de possuir.

Os benefícios para as pessoas que poderão usar esses dados são gigantes. Se o usuário puder cumprir com os benefícios seguintes, você cumpriu com o dever da primeira estrela:

- Usuário pode procurar pela informação;
- Usuário pode imprimir;
- Pode guardar localmente na sua máquina, pendrive etc.;

- Pode transferir esses dados para qualquer outro dispositivo próprio ou de outra pessoa;
- Pode modificar esses dados da maneira que achar necessária;
- Pode compartilhar esses dados com qualquer um, da maneira que quiser;
- Não precisar explicar repetidas vezes que terceiros podem usar esses dados da maneira que bem entender.

Estrela 2

Seu conteúdo está disponível em um formato estruturado para que as máquinas consigam ler? Por exemplo, em formato de Excel em vez de um formato de imagem.

Tudo o que pode ser texto, disponibilize como texto. Publicar uma imagem com uma tabela é muito pior do que publicar um arquivo Excel, aberto, para que qualquer um consiga baixar, manusear, reutilizar ou simplesmente visualizar. Além dos benefícios da primeira estrela, o usuário também poderá:

- Modificar os dados, editando-os com softwares abertos ou proprietários, de forma que ele possa visualizar, agregar e editar esses dados;
- Exportar esse arquivo em outros formatos de dados estruturados, facilitando a publicação e compartilhamento com outros meios de consumo de dados.

Estrela 3

Seus dados estão disponíveis em um formato não proprietário, como CSV, por exemplo. É importante que o usuário não precise usar softwares proprietários específicos para manipular os dados que você publicou. Ele pode pegar um CSV e abrir no Google Spreadsheet, no Excel, no Libre, no Numbers ou em qualquer outro software que entenda esse formato.

Da mesma forma que não seja necessário a utilização de qualquer plugin, add-on ou algo do tipo. Lembre-se de que esses dados precisam ser acessíveis e fáceis de publicar. Lembra-se de que a regra de ouro dos tempos do Vannevar Bush era “guardar e consultar”.

Estrela 4

Aqui você já está em um nível quase modelo. Você está publicando seus dados, cumprindo todos os requisitos anteriores e mais: usa padrões abertos do W3C (como RDF, JSON-LD etc.).

Ter dados abertos em formatos padronizados permite:

- Adicionar links no seu dado de um lugar para outro, tanto na Web quanto localmente;
- Usuário pode reutilizar apenas partes dos dados;
- Pode combinar dados com outros dados de formatos compatíveis;
- Os dados têm uma URI para identificá-los. Logo, duas coisas podem ter a mesma URI de forma intencional;
- Como o dado tem uma URI única, o usuário pode guardar;
- Outras pessoas que publicaram dados em outros lugares podem inserir links diretamente para sua publicação.

Estrela 5

Você cumpriu todos os requisitos anteriores e mais: seus dados contêm links para outros recursos de informação para criar um contexto. É importante que eles estejam disponíveis na internet, de forma que seja possível relacioná-los com outros dados, formando uma relação global com estes.

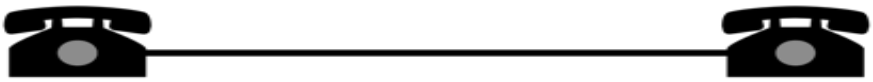
Contudo surge um problema: a indisponibilidade eventual desses dados. Conhece o **Erro 404**? Pois é. Esse é o grande problema da Web hoje. Atualmente, o próprio usuário deve lidar quando chega

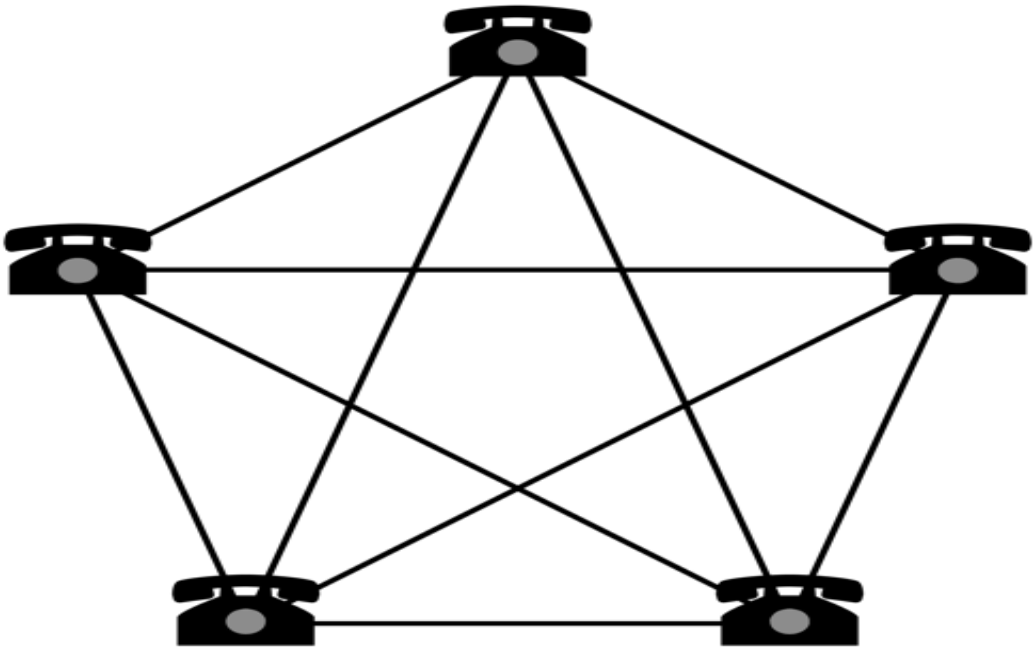
em um beco sem saída: ou ele volta para a página anterior e continua procurando, ou desiste.

Como alguém que publica dados na internet, você deve ter certeza de que seus dados possam ser descobertos na Web. Você também deve investir tempo, e talvez dinheiro, para que seus dados estejam *linkados* com outros dados e, principalmente, tentar se precaver com links quebrados.

> Em economia e negócios, um efeito de rede (também designado externalidade de rede, ou procura de economias de escala) é o efeito que um utilizador de um bem ou serviço tem sobre o valor do produto para outros utilizadores. Quando o efeito de rede estiver presente, o valor de um produto ou serviço depende do número de utilizações de outras pessoas (https://pt.wikipedia.org/wiki/Efeito_de_rede).

Nesse nível, você já deve estar se beneficiando do “efeito de rede" causado pelo relacionamento dos dados entre diversas fontes na internet.





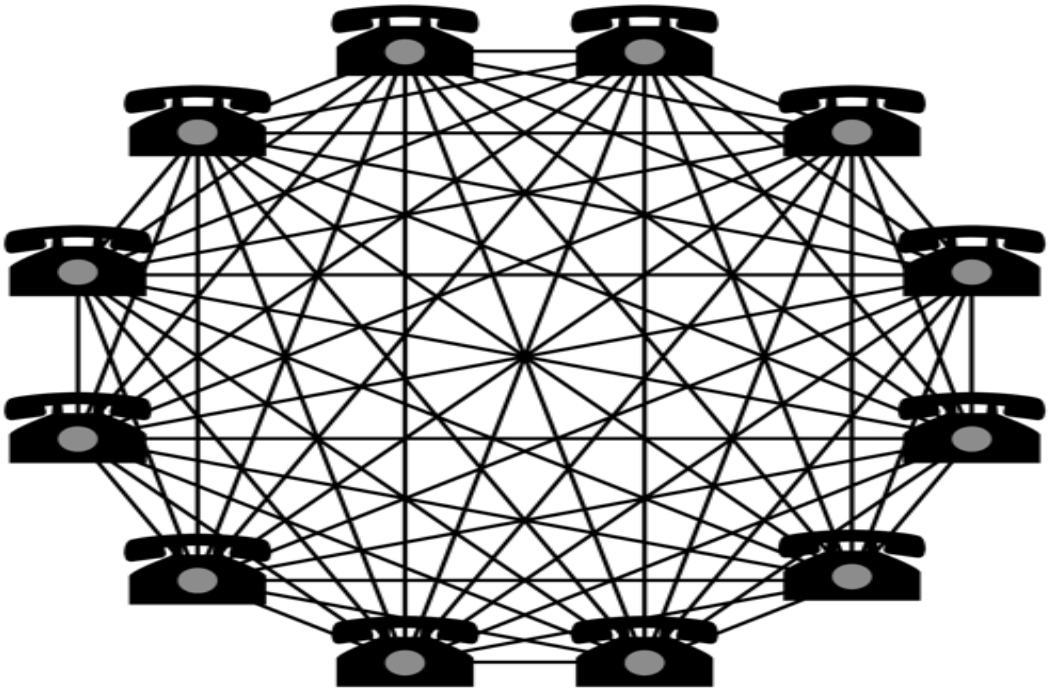


Figura 4.4: Fonte: https://commons.wikimedia.org/wiki/File:Metcalfe-Network-Effect.svg

4.3 Linked Data API

Open Data Certificate

A criação dessas estrelas é algo muito simbólico e conceitual sobre ter dados realmente abertos na internet. Mas lembre-se de que a confiança na fonte e na veracidade desses dados é uma premissa muito importante para que a Web Semântica funcione de verdade. A NASA não quer usar dados errados para fazer suas pesquisas, nem empresas em seus negócios.

E para ajudar na divulgação do conceito de Linked Data e da Web Semântica, foi criada uma certificação aberta e online, chamada de **Open Data Certificate**, pelo **Open Data Institute**. O Open Data Certificate (https://certificates.theodi.org/en) é uma autocertificação, isto é, por meio de uma ferramenta livre, as empresas, governos e outras instituições submetem seus dados para verificar o cumprimento dos níveis de estrelas que vimos anteriormente.

Esse site é mais indicado para empresas ou governos que querem certificar seus arquivos de dados que serão publicados na internet para consulta e relacionamento de outras fontes de informação. Um exemplo bizarro: lá você vai encontrar arquivos da NASA, que descrevem dados sobre a estrutura e composição vegetal da Amazônia (https://certificates.theodi.org/en/datasets/47279) ou dados sobre os níveis de concentração de carbono e nitrogênio no cerrado brasileiro (https://certificates.theodi.org/en/datasets/94390/certificate).

4.4 Sobre o conceito de dados abertos

Imagine se os governos, ONGS, empresas, grupos independentes e todas as outras pessoas do mundo deixassem dados importantes, de interesse público, abertos. Pense nas possibilidades e nas respostas que a humanidade encontraria ao cruzar esses dados.

> A Web é um lugar em que todos os dados importantes circulam. Assista esse vídeo do Hans Rosling para mais deste assunto. São apenas 19m50 de vídeo que vão lhe ajudar a entender o que eu quero dizer: http://bit.ly/hans-rosling-best-stats-ted.

Hans Rosling conseguiu fazer essa apresentação, mostrada anteriormente, apenas porque os dados estavam abertos. Mas não foi fácil. Os dados estavam em diferentes formatos, páginas em sites escondidos, mal indexados etc. Ele não conseguiu tudo isso buscando apenas no Google. Mas embora tenha sido difícil encontrar esses dados, todos eles estavam disponíveis de alguma forma.

Ter os dados abertos em formatos acessíveis é fazer com que as máquinas façam o trabalho pesado para nós. O Hans Rosling conseguiu esses dados, cruzou todos eles e deu respostas para perguntas importantes para a sociedade e outros pesquisadores.

Dados são chatos. Dados sozinhos não nos dizem nada. Na sua grande maioria, são um monte de números espalhados, que sozinhos não dizem absolutamente nada. Contudo, eles dirigem grande parte das suas decisões diárias e, principalmente, as grandes decisões tomadas no mundo todo. A economia do mundo todo é dirigida por dados. O que Hans Rosling fez foi misturar números que não diziam nada para muitas pessoas, transformando esses números em informação. Informação essa que, por sua vez, vai ajudar empresas a criarem produtos e outros negócios para ajudar mais pessoas.

Esse é o real conceito do Linked Data: expor dados crus, não adulterados, para que o mundo consiga usá-los de formas

diferentes. É por isso que empresas como Apple, Google, Amazon, Microsoft e outras são tão fanáticas pelos dados que geramos. Elas querem saber desde como dormimos até quanto gastamos diariamente. Todos dias geramos dados que, quando sozinhos, não dizem muita coisa, mas que, se relacionados, dizem muito sobre nós. São dados crus que damos a essas empresas para que elas tomem decisões por nós. Mas o importante é saber que essas empresas usam os dados gerados por você para o benefício delas. Claro, elas fazem apps maravilhosos sobre saúde, finanças etc., porém, qual o real benefício que você ganha?

A mesma bagunça acontece quando os governos não disponibilizam apropriadamente seus dados. Nós temos a sorte de ter um governo que libera muitos dados por meio de dois sites, o primeiro deles é o Portal da Transparência (http://transparencia.gov.br/). Nesse site, é disponibilizada uma série de informações como gastos, pagamentos, transferências e gastos com programas sociais, em formato CSV. Mas só liberar esses dados não quer dizer muita coisa. Você precisa disponibilizar em um formato acessível. Embora seja muito usado por aí, CSV não é a melhor maneira para disponibilizar dados.

Contudo, o Governo Federal disponibiliza mais dados em um outro site, muito melhor, chamado Portal Brasileiro de Dados Abertos (http://dados.gov.br/). Lá você encontra dados sobre uma série de órgãos do governo brasileiro em diversos formatos.

O Portal Brasileiro de Dados Abertos é a ferramenta disponibilizada pelo governo para que todos possam encontrar e usar os dados e as informações públicas. O portal preza pela simplicidade e organização para que você possa encontrar facilmente as informações que precisa. O portal também tem o objetivo de promover a interlocução entre atores da sociedade e o governo para pensar a melhor utilização dos dados em prol de uma sociedade melhor.

O Governo dos Estados Unidos também tem suas iniciativas. Há um site chamado *Data.gov* (https://www.data.gov), em que são divulgados todos os dados do país. Além disso, você consegue cruzar dados, criando gráficos para comparação. O país também tem um site chamado *Project Open Data*:

> *"Data is a valuable national resource and a strategic asset to the U.S. Government, its partners, and the public. Managing this data as an asset and making it available, discoverable, and usable – in a word, open – not only strengthens our democracy and promotes efficiency and effectiveness in government, but also has the potential to create economic opportunity and improve citizens' quality of life"*
>
> Fonte: https://project-open-data.cio.gov

A Casa Branca desenvolveu esse site para divulgar e ajudar agências do governo a adotar uma política de dados abertos. Além disso, esse é um projeto *open source*, aberto para qualquer um fazer fork no GitHub (https://github.com/project-open-data/project-open-data.github.io/)! Eles motivam que os governos abram seus dados no formato JSON-LD, para que todos os dados possam ficar disponíveis, localizáveis e usáveis para qualquer um no mundo.

4.5 Definição de aberto

Mas o que é exatamente ter um dado aberto? Existe um site chamado *Open Definition* (http://opendefinition.org/od/2.0/pt-br/), que ajuda a definir exatamente o significado de "aberto" quando diz respeito a conhecimento.

> O conhecimento é aberto se qualquer pessoa está livre para acessá-lo, utilizá-lo, modificá-lo, e compartilhá-lo — restrito, no máximo, a medidas que preservam a proveniência e abertura.

O site *Open Definition* foi inicialmente derivada da *Open Source Definition* (https://opensource.org/), que é formado pela comunidade de software livre e que ajuda a definir o que é realmente software não proprietário. Este, por sua vez, foi derivado da *Debian Free Software Guidelines* (http://www.debian.org/social_contract).

Se o princípio dos dados abertos for realmente seguido por pessoas, governos e empresas, a Web pode *definitivamente* se transformar em uma plataforma interoperável, na qual todos podem ter acesso à grande massa de conhecimento gerada pela humanidade e também pelas máquinas.

4.6 A stack de tecnologias da Web Semântica

O termo Web Semântica muitas vezes se refere a um conjunto de tecnologias. Essas tecnologias fazem com que a Web de Dados seja possível, possibilitando o relacionamento entre os dados disponíveis na internet. Para ilustrar o relacionamento dessas tecnologias, existe uma ilustração feita pelo Tim Berners-Lee, que tem o apelido de *Bolo de Noiva*, ou *Semantic Web Stack* (ou *Semantic Web Layer Cake*):

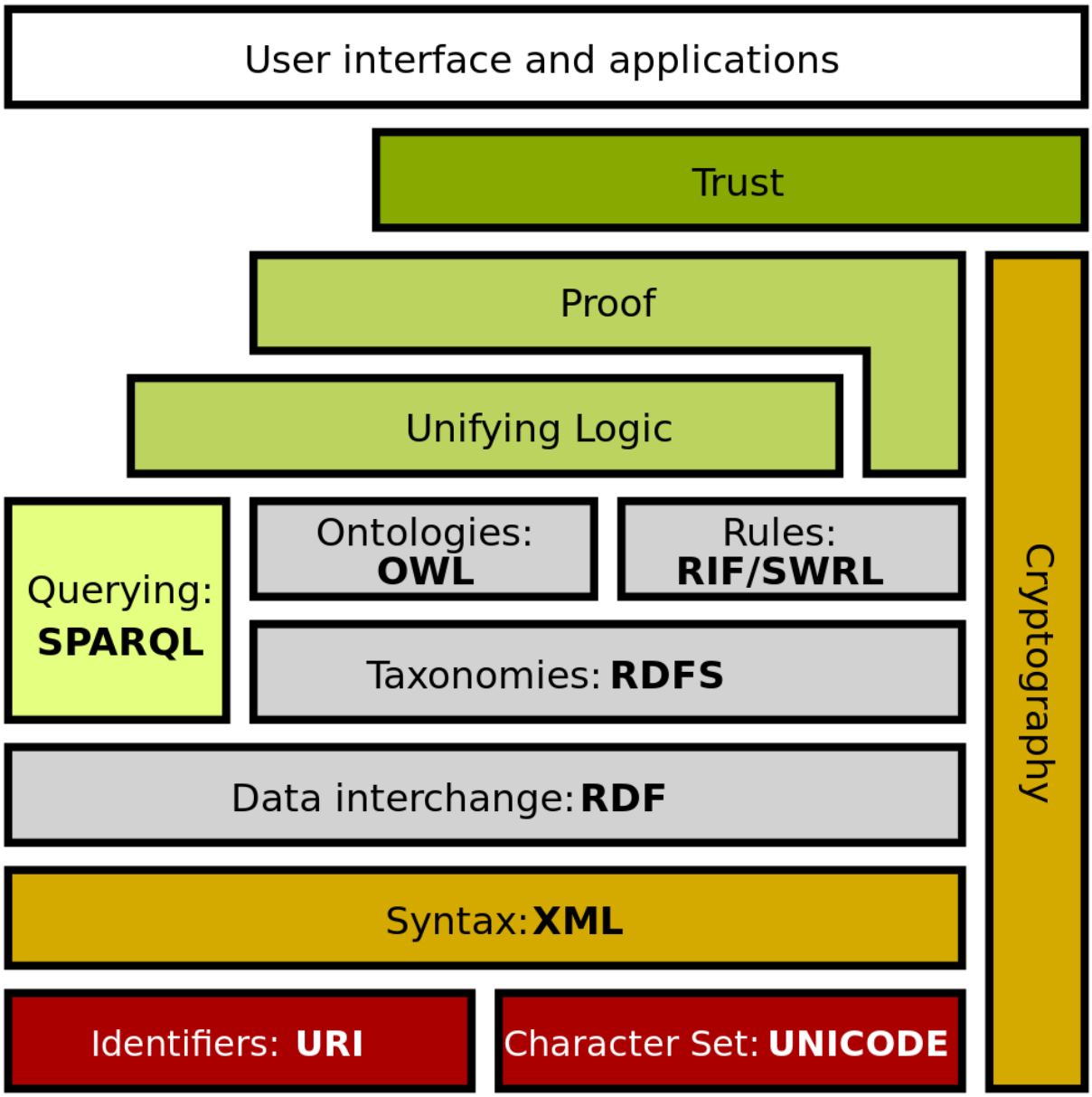


Figura 4.5: Fonte: https://en.wikipedia.org/wiki/Semantic_Web_Stack

Cada camada na imagem explora e utiliza as capacidades das camadas abaixo. Isso mostra como as tecnologias estão organizadas para fazer a Web Semântica possível e também mostra como **a Web Semântica é uma extensão**, e **não uma substituição** da Web atual.

As tecnologias da Web Semântica têm um potencial imenso para endereçar as necessidades de relacionamento de dados, mantendo

esses dados fáceis de descobrir e processar, tanto por humanos quanto por máquinas, que é o que chamamos de Linked Data. Além desse conceito, as tecnologias usadas são bem importantes para que tudo se torne real. Neste livro, não vou abordar todas as tecnologias listadas ali, mas quero que você entenda boa parte sobre o papel e a importância que cada uma representa para a Web de Dados.

Perceba que, na imagem, as tecnologias foram colocadas umas em cima das outras. Isso é porque as de cima dependem das de baixo. URI e Unicode (que veremos nos capítulos a seguir) servem como base para tudo. Isso tem um motivo: Unicode é o padrão de caracteres que as outras tecnologias se basearão, como as URIs, que são expressadas usando base de caracteres latina. Mesmo assim, o Unicode é essencial para representar e manipular os textos dos documentos na internet, sem o bloqueio da necessidade de múltiplas tabelas de caracteres para cada idioma.

As URIs entram com o papel de identificar e localizar os *resources* (recursos) na internet. Todas as fontes de informação disponíveis na internet devem ter uma URI única que a identifique e localize. Sem isso, não há como relacionarmos informações. Sem relacionar informação, não existe Web Semântica.

A sintaxe XML é a linguagem base e padrão de marcação do W3C para distribuição e estruturação de dados na internet. O XML é usado ainda hoje para distribuir informação como nos Feeds RSS/Atom, por exemplo. Como a essência da Web Semântica é conectar dados, nós precisamos referir várias fontes em um mesmo documento e, para isso, existem os Namespaces XML.

Baseado em XML, o RDF entra como o framework para criar os estados de relacionamentos dos dados em forma de triplas. Isso acontece assim como o RDFS, que provê um vocabulário básico para o próprio RDF, possibilitando a criação de hierarquias de classes e propriedades.

Em cima do RDF e do RDFSchema, temos a camada OWL, que significa *Web Ontology Language*. O OWL estende o RDFS, adicionando mais formas para descrever os relacionamentos dos dados. Perceba que ali do lado existe um quadradinho do SPARQL. O SPARQL é uma linguagem de query, usada para extrair informações das aplicações.

Não vou abordar aqui nada sobre as camadas *Unifying Logic* (ou *Logic Framework*, em algumas versões da Stack), *Proof* ou *Trust*. Elas ainda não foram desenvolvidas, como é o caso da **Trust** (que seria a camada que confere a confiança entre os recursos). Como uma das bases da Web Semântica é conectar dados, como alguém que precisa consumir os dados de um outro site pode saber que a fonte daquele dado e o dado propriamente dito é confiável? Essa camada responderia esta pergunta, garantindo que as informações fornecidas são válidas e qual o grau de confiança dessa fonte de informação.

Vale lembrar que todas essas tecnologias são a base para a utilização de outras tecnologias. Por exemplo, se um site de conteúdo quiser publicar seus dados, muito provavelmente esse site fará isso expondo um JSON-LD, que é uma serialização do RDF, por exemplo.

Mas nós vamos usar exatamente todas as tecnologias que o W3C instituiu nessa stack? A resposta é um estridente **não**.

A imagem adiante mostra como as tecnologias da Web Semântica se conversariam. A parte esquerda, que fala sobre a Web, mostra como uma URI se torna uma representação de um documento, usando um HTTP. Por exemplo, ela poderia ser uma string de bits com um MIME type específico. Então, isso é parseado em um XML (calma, mais para a frente você entenderá que não vamos precisar usar XML de verdade) e então em RDF, para produzir um grafo ou uma formula lógica. Na direita, a parte Semântica mostra como um grafo RDF contém referências para a URI.

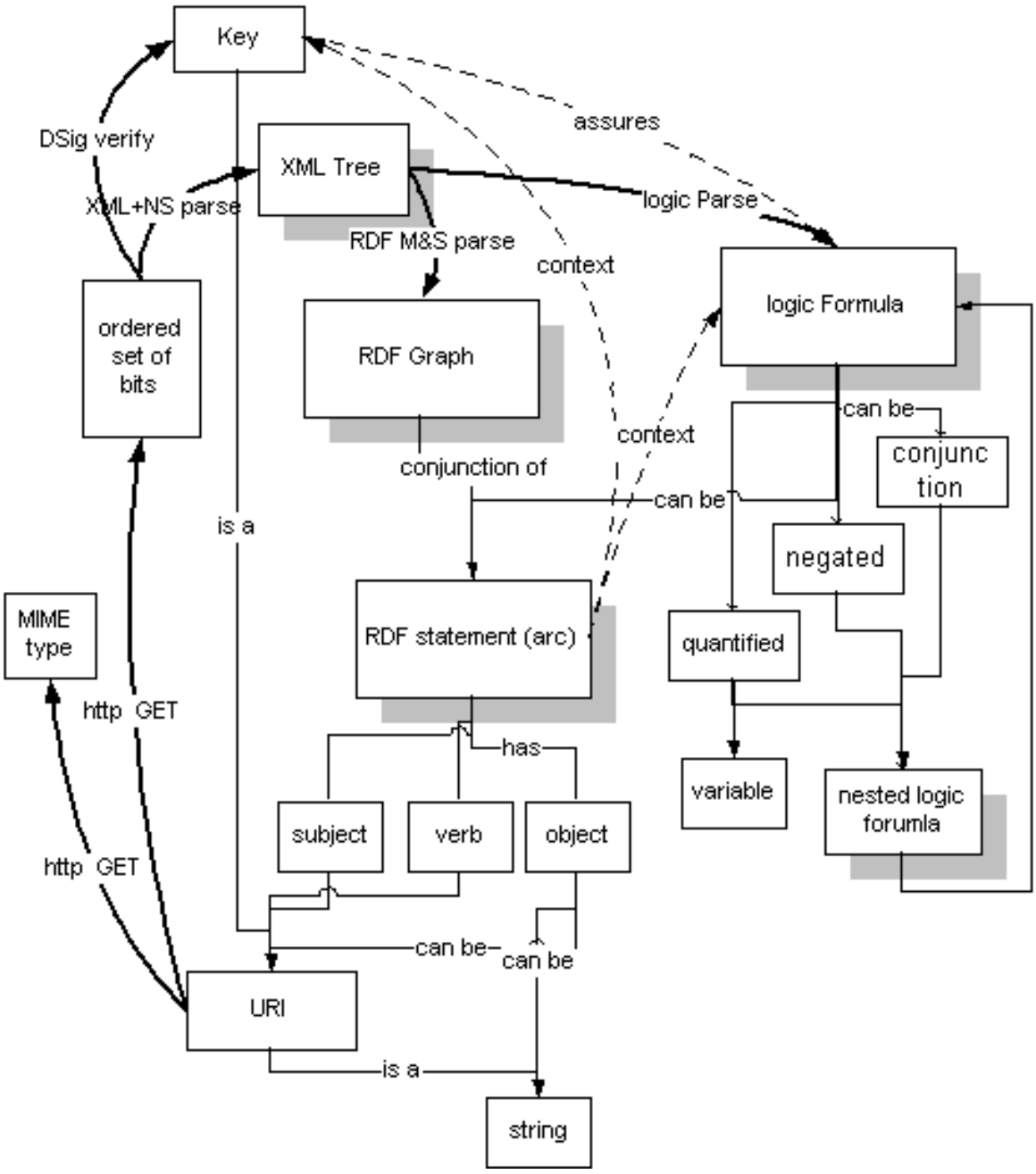


Figura 4.6: Fonte: https://www.w3.org/DesignIssues/Semantic.html

4.7 Não acredite no stack original da Web Semântica

Manu Sporny (http://manu.sporny.org) é um dos caras que está envolvido nessa coisa toda de Web Semântica e dados estruturados, além de ser um dos criadores do JSON-LD. Ele fez um artigo intitulado **JSON-LD and Why I Hate the Semantic Web** (http://manu.sporny.org/2014/json-ld-origins-2/), no qual ele mostra toda a confusão que é o stack original idealizado pelo W3C.

Ele defende que várias tecnologias usadas para "pregar" a Web Semântica são totalmente desconhecidas pelo grande público dos desenvolvedores, como por exemplo, o Turtle, o SPARQL e até o RDF. Eu nunca tinha ouvido falar sobre Turtle ou SPARQL até começar a escrever este livro e, provavelmente, você nunca deve ter ouvido falar também. E é exatamente por isso que o Manu Sporny diz que, por causa desse desconhecimento por parte dos devs, a Web Semântica se torna algo complicado, burocrático e demorado para ser aplicado.

Além de que algumas tecnologias são ruins por natureza, como é o caso do XML, que não é o formato mais adorado pelos devs para distribuição de dados, porém, é a base principal do RDF. Este é uma linguagem para representar e estruturar dados entre máquinas (veremos um pouco sobre RDF, Turtle e SPARQL mais à frente, fique tranquilo).

O autor sustenta muito bem a ideia de que as tecnologias envolvidas deveriam ser mais simples e baseadas nas tecnologias que já são largamente conhecidas, como é o caso do JSON, que serve como base para o JSON-LD (um formato de distribuição de dados, concorrente do RDF, que também veremos mais à frente). Eu concordo com ele em várias partes e confesso que, na jornada de escrever este livro, eu me senti perdido várias vezes com a quantidade de tecnologias envolvidas.

Sporny também diz que, em boa parte do tempo que ele está envolvido nesse assunto, nunca precisou usar RDF/XML, N3, NTriples, Turtle ou SPARQL para fazer com que seus sistemas se tornassem "semânticos". Esse foi um dos motivos pelo qual ele

iniciou a criação e a divulgação do JSON-LD. A boa notícia é que você também não precisará usar todas essas tecnologias, nem precisará saber que elas existem para entrar de cabeça na Web Semântica. Exatamente por isso que eu não abordei todas no livro, já filtrando uma grande quantidade de besteiras para você.

Seus dados precisam estar disponíveis na internet de forma que alguém (humano ou máquina) consiga acessá-los, manipulá-los e relacioná-los sem nenhum limite. O problema é como fazer isso e em qual formato essas informações estarão disponíveis. Nos próximos capítulos, vou abordar as tecnologias citadas. Em algumas me aprofundarei mais, em outras menos.

Embora haja tecnologias que você não precisa conhecer a fundo, a Web Semântica é baseada em tecnologias sólidas e que você precisa estar familiarizado com elas; não apenas para entender a Web Semântica, mas para entender sobre o seu trabalho diário. Vamos começar entendendo um pouco mais sobre duas tecnologias que formam a estrutura da internet: Unicode e URI.

4.8 Para ler mais

- Vídeo do Tim Berners-Lee falando sobre LD e Web Semântica — https://www.youtube.com/watch?v=ga1aSJXCFe0
- Vídeo sobre a Web de Dados — https://www.youtube.com/watch?v=HeUrEh-nqtU
- As 5 Estrelas dos Dados Abertos — http://5stardata.info/pt-BR/
- W3C e o Linked Data — https://www.w3.org/standards/semanticweb/data
- BERNERS-LEE, Tim. *Semantic Web Roadmap.* 1998. Disponível em: https://www.w3.org/DesignIssues/Semantic.html.

- BERNERS-LEE, Tim. *Linked Data.* 2006. Disponível em: http://www.w3.org/DesignIssues/LinkedData.
- BIZER, Chris; CYGANIAK, Richard; HEATH, Tom. *How to Publish Linked Data on the Web.* 2007. Disponível em: http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/.
- KANTOLA, Juhani. *Is your Linked Data 5 Star Rated?* 2013. Disponível em: http://www.citysdk.eu/is-your-linked-open-data-5-star-rated/.
- MILLER, Eric; SWICK, Ralph. *An Overview of W3C Semantic Web Activity.* 2003. Disponível em: http://onlinelibrary.wiley.com/doi/10.1002/bult.280/full.
- RAIMOND, Yves; SMETHURST, Michael. *A skim-head introduction to linked data.* 2005. Disponível em: http://www.bbc.co.uk/blogs/radiolabs/s5/linked-data/s5.html.

Capítulo 5
Unicode e URI

> *The most important thing that was new was the idea of URI-or URL, that any piece of information anywhere should have an identifier, which will allow you to get hold of it.* **— Tim Berners-Lee**

5.1 A base da internet

O funcionamento da World Wide Web, ou apenas Web, é muito simples e talvez você já saiba, ao menos por alto, como ela funciona. Contudo, gostaria de dar mais detalhes sobre esse funcionamento, bem como sobre suas tecnologias. Isso será interessante para entendermos os próximos conceitos sobre a Web Semântica.

A Web é o lugar no qual publicamos e produzimos informação, em uma determinada fonte. Essa fonte de informação pode ser relacionada a outras fontes e pode ser identificada por um padrão global que chamamos de URI (veremos mais detalhes a seguir).

A Web consiste em três arquiteturas base: identificação, interação e formato. O W3C dá um exemplo para entendermos isso (https://www.w3.org/TR/webarch/): uma pessoa está planejando uma viagem para Oaxaca, no México, e vê em uma revista o seguinte endereço **http://weather.example.com/oaxaca**, que mostra informações sobre o clima do lugar. Ela sabe que isso é o endereço de um site e que, para visitar, ela precisa de um browser. Quando essa pessoa coloca o endereço no browser, alguns passos básicos acontecem:

1. O browser reconhece o endereço digitado;
2. O browser detecta o protocolo como sendo `http` e separa as ações necessárias para esse tipo de protocolo;
3. O servidor responsável pelo endereço passa informações no primeiro request de resposta;
4. O browser, por sua vez, interpreta essa resposta, identifica como HTML pelo servidor, e inicia o processo para mostrar o conteúdo;
5. O browser mostra a informação processada pronta para que o usuário possa interagir.

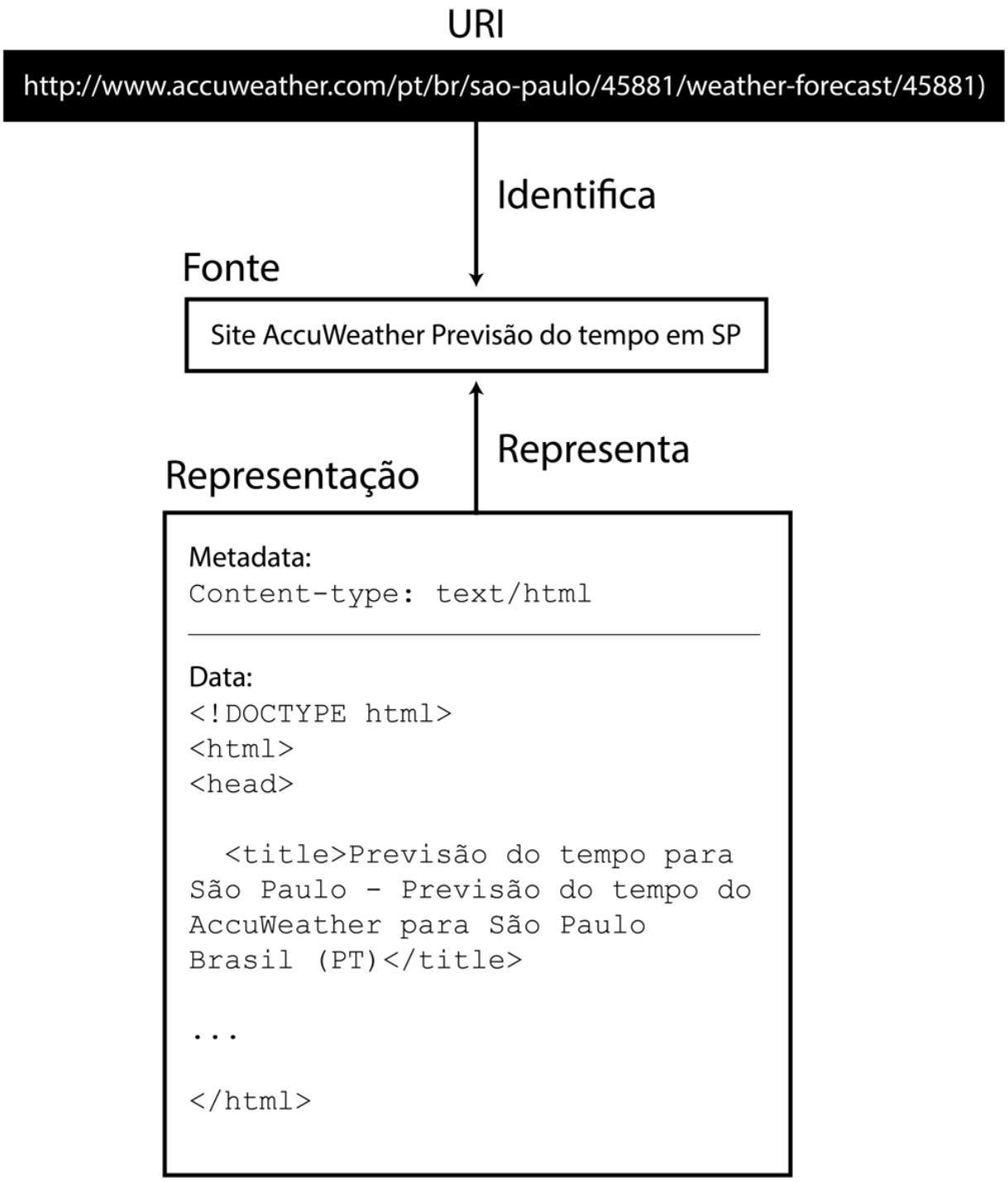
6. **Identificação**: o browser detecta e identifica a fonte de informação usando a URI (no formato de URL) que a pessoa digitou no browser. Se fosse na vida real, esse endereço poderia ser **http://www.accuweather.com/pt/br/sao-paulo/45881/weather-forecast/45881**, que mostra informações sobre o tempo na cidade de São Paulo. Esse endereço é uma URI, e está servindo para identificar essa fonte em específico.

7. **Interação**: logo depois, o browser se comunica usando o protocolo padrão, que habilita a troca de mensagens entre cliente e servidor. Quando o usuário insere uma URI ou interage com um link, este fala para o browser que está querendo se conectar com aquela fonte, nesse caso pelo protocolo `http`, enviando uma requisição `GET` para o servidor, via porta 80 do protocolo TCP/IP.

8. **Formato**: a maioria dos protocolos web enviam e recebem uma resposta de uma ou mais mensagens dos servidores, contendo dados e metadados. Cada um desses protocolos (FTP, SMTP, POP, IMAP etc.) tem seus limites para representar esses dados. O HTTP, por exemplo, usa os cabeçalhos **Content-Type** e **Content-Encoding** para identificar o formato dessa representação de dados.

   Nesse nosso exemplo, a representação de dados em questão é a HTML. Se fosse XML ou até mesmo XHTML, ela seria

identificada pelo header de **Content-Type** como uma `application/xhtml+xml` . Quando o browser recebe essas informações, ele sabe como interpretar esses dados de acordo com a sua especificação. O browser precisa saber disso, se não ele interpreta os dados de uma maneira errada.



Quando os padrões web estavam se popularizando, tanto no Brasil quanto no mundo, era muito comum que as pessoas usassem código XHTML Strict em vez de HTML 4.01, por ser um formato mais restrito a erros de sintaxe. Como o XHTML também é uma extensão do XML, era necessário servir nosso código XHTML com o **mime type** (http://www.webstandards.org/learn/articles/askw3c/sep2003/) de XML `application/xhtml+xml`.

Todo mundo servia como um HTML normal usando o mime type `text/html`. Totalmente errado. Embora o browser conseguisse renderizar bem os sites por causa de uma série de regras de *fault-tolerance*, o tipo de conteúdo servido pelos sites era totalmente equivocado. Se não quisesse mudar como o servidor estava servindo os arquivos `.html`, era só declarar a tag `meta` assim:

```
<meta http-equiv="Content-Type" content="application/xhtml+xml;
charset=utf-8"/>
```

Isso não causava grandes problemas para a internet. Talvez, se tivesse quebrado alguma coisa, isso teria virado regra e todo mundo teria servido seus arquivos da maneira correta. Mas tudo bem, estamos em 2017 agora (bem, depende de quando você está lendo isso).

Recursos

Para publicar dados na internet, nós primeiro precisamos identificar o que o W3C chama de *items of interest*. Esses itens são chamados de recursos e são basicamente divididos em dois tipos: *recursos de informa ão* e *recursos de não-informa ão* (do inglês *non-information resources*).

Recursos de informa ão são aqueles dados que conhecemos na web tradicional como imagens, texto, documentos, vídeos, sons e outros tipos de arquivos de mídia. Mas existe uma série de outras coisas sobre as quais queremos compartilhar informações, como

pessoas, produtos, lugares, empresas, conceitos (operações matemáticas, conceitos científicos etc.), e estes, por sua vez, são chamados de *recursos de não-informa ão*. Para ficar fácil de entender o que é cada um, pense que qualquer objeto real, que exista fora da web, é um *recurso de não-informa ão*.

Essa distinção é muito importante para o contexto de Linked Data, porque mostramos para as máquinas sobre qual tipo de dados exatamente elas estão lidando e como elas podem usar esses dados. Um sistema de indexação de imagens, por exemplo, poderá tratar a imagem de uma pessoa diferente do que trata a imagem de um gráfico. As informações que serão relacionadas com cada tipo de recurso serão diferentes.

Cada um dos recursos precisa ser identificado de alguma forma para evitar ambiguidades. A identificação também serve para dar algum contexto sobre o relacionamento dos assuntos conectados. Esses recursos são identificados por URIs baseadas no protocolo HTTP.

O W3C restringiu o conceito de Linked Data para usar apenas URIs baseadas em HTTP, evitando outros tipos de URIs como URNs (https://www.w3.org/TR/uri-clarification/) e DOIs. Isso é bom, porque as URIs baseadas em HTTP servem para identificar, mas também localizar, como é o caso das URLs. Vamos ver mais detalhes a seguir.

5.2 Por que o Unicode é importante?

Unicode é um padrão internacional de codificação de conjuntos e caracteres. Isso permite que toda a linguagem humana, escrita ou lida, usada na web tenha um padrão unificado.

Eu não vou explicar aqui como o funciona todo o processo de quando apertamos uma tecla no teclado e uma letra aparece

magicamente na nossa tela. Mas resumindo bastante: o computador usa uma tabela para localizar os caracteres que ele deve mostrar na tela. Quando você aperta uma tecla, o sistema operacional entende a tecla apertada, procura o código relativo àquela tecla e mostra na tela o valor correspondente.

O ponto é: cada país tem um idioma diferente e nem todos os caracteres para expressar alguns desses idiomas existem nessa tabela. Logo, muitas tabelas de caracteres foram criadas para suprir essa necessidade. Uma tabela para a China, outra para os EUA, outra para o Japão, Rússia etc. O problema disso é que, se eu entro em um site ou tenho um documento escrito em japonês, pode ser que meu computador não tenha essa tabela japonesa.

Claro, todos os sistemas operacionais hoje em dia têm todas as tabelas de caracteres disponíveis para ocasiões assim. Mas isso gera outro problema: quando fazemos um documento, temos de especificar que estamos usando determinada tabela de caracteres para que os sistemas operacionais possam interpretar corretamente cada um dos documentos. Isso acontece corriqueiramente quando escrevemos um documento HTML, por exemplo. Nós sempre especificamos uma tag `meta` no `head` do documento dizendo qual é a tabela de caracteres usada para fazer aquele documento:

```
<meta charset="iso-8859-1">
```

Essa é uma tabela de caracteres da ISO/IEC (https://en.wikipedia.org/wiki/ISO/IEC_JTC_1) para os caracteres usados no idioma latino (https://en.wikipedia.org/wiki/ISO/IEC_8859-1). É a que usávamos quando fazíamos qualquer coisa para a internet no idioma português.

Esse conjunto de caracteres tem o nome de **Charset**. Um conjunto de caracteres (Charset) consiste em ter:

1. Repertório com caracteres;
2. Uma posição de referência para cada um dos caracteres no repertório.

Cada caractere é identificado e localizado por um código de posição. Por exemplo, na tabela ASCII, as posições 65, 66 e 67 se referem às letras A, B e C, respectivamente. A tabela ASCII tem 127 caracteres. Os primeiros 32 caracteres são códigos de controle para periféricos, como impressoras. Os códigos do 32 ao 127 são chamados de caracteres de impressão e representam números, letras, pontuações e símbolos. O caractere 127 representa o comando `DEL` . A versão Microsoft Windows Latin-1 estende esse número para 256 caracteres. A seguir, veja a tabela com os 127 caracteres:

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[ENG OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

A tabela de caracteres ASCII (Código Padrão Americano para Intercâmbio de Informações) foi a primeira usada em larga escala. O computador foi desenvolvido nos Estados Unidos. No vocabulário americano, não existem acentos, além disso, era um código de 7 bits, e não 8. Ou seja, em vez de 256 posições, a tabela ASCII tinha apenas 128 — como você sabe, tudo nos computadores são um grupo de zeros (0) e uns (1) chamados de bits. Esses zeros e uns formam grupos de oito em oito que são chamados de bytes e

representam um número entre 0 e 255. Assim como as imagens, áudios, vídeos, programas e tudo o que temos nos sistemas de hoje, os caracteres que aparecem na sua tela são grupos de zeros e uns.

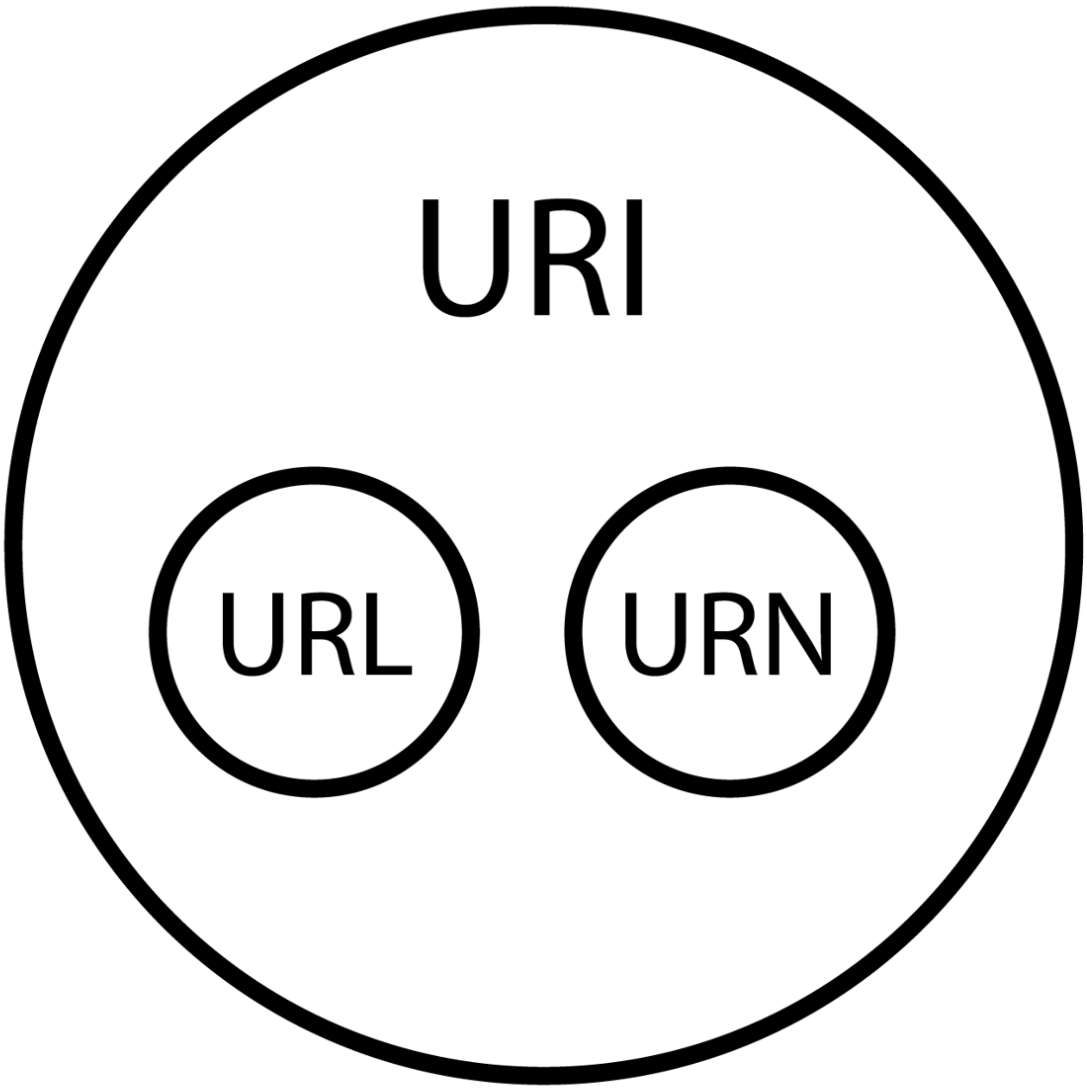
Outro problema é que cada tabela tem seus próprios códigos para cada caractere, tornando os sistemas de codificação conflitantes. Isso quer dizer que o código que representa a letra **A** pode ser diferente entre as outras tabelas.

Para acabar com a confusão da existência de milhares de tabelas de caracteres espalhadas em todas as partes do mundo, um esforço foi feito para criar uma tabela única, em que cada caractere tem um código específico, não importando a plataforma, o idioma ou o sistema usado. Esse novo sistema chama-se Unicode.

Logo, o Unicode permite que o mundo inteiro consiga produzir e acessar documentos sem o problema de incompatibilidade de caracteres entre as duas pontas. Todos os padrões de tecnologias como JavaScript, XML e outras se baseiam no Unicode para funcionarem. A Web Semântica precisa de um padrão para a troca de informações universal, por isso ter uma tabela única com todos os caracteres é importante, possibilitando a troca de informação, não importando a origem da requisição da outra ponta.

5.3 O que são as URIs

URI (ou *Universal Resource Identifier*) é uma simples string, que serve para identificar um recurso na internet. É como se fosse um RG de uma fonte de informação, fazendo com que determinada fonte seja identificável e única. Uma URI pode identificar, localizar ou fazer as duas tarefas ao mesmo tempo.



As URIs são apenas uma sequência de caracteres da tabela do alfabeto básico latino, incluindo dígitos e alguns caracteres especiais. Este é um dos motivos pelo qual a tabela de caracteres que você viu sobre Unicode é importante. A interpretação da URI depende apenas dos caracteres usados, e não em como esses caracteres são representados no protocolo de rede.

A RFC dá algumas considerações sobre o design das URIs:

- A URI é uma sequência de caracteres que nem sempre são representados como uma sequência de octetos.

- Uma URI pode ser transcrita de uma fonte que não seja de rede, como de um teclado, mas provavelmente por meio de um computador. Tendo em consideração as limitações impostas pelos teclados (ou dispositivos parecidos) através dos idiomas e localidades.
- Deve ser fácil para uma pessoa memorizar uma URI, sendo uma boa prática usar formas que facilitem essa lembrança.

Uma parte importante é que há casos nos quais, para dar significado à URI, é necessário usar caracteres que não sejam comuns ou que não podem ser usados em alguns dispositivos por causa de suas limitações. Nesse caso, a possibilidade de transcrever a URI em um dispositivo é mais importante do que seu significado.

É interessante notar que uma URI não precisa ser referência para uma fonte acessível. A URI, por si só, garante apenas identificação de uma fonte. O acesso aos dados dessa fonte está associado por um elemento de protocolo, formato de dados ou linguagem natural em que o texto aparece. Dada uma URI, o sistema vai tentar executar uma série de operações. Essas operações são executadas pelo protocolo, como por exemplo, os métodos HTTP (`GET` , `POST` , `PUT` , `DELETE` etc.), que enviam uma ação que deverá ser executada por meio do recurso especificado.

As URLs fazem parte do padrão URI. Existe muito desenvolvedor por aí que acha que URI é a mesma coisa que URL, mas não é. Toda URL é uma URI, mas nem toda URI é uma URL. A URL (*Unified Resource Locator*), como bem sabido, é um endereço, de onde se encontra um recurso na internet. É como se fosse o endereço de um arquivo no seu computador, só que de forma mais ampla.

Como existem vários protocolos que usam a internet como base, a URL não usa apenas o HTTP, mas pode ser uma URL do protocolo FTP ou SMB, por exemplo. O ponto é que a internet é baseada no

protocolo HTTP, por isso o conceito de Linked Data se aplica apenas às URIs baseadas em HTTP.

A seguir, veja algumas variações de URIs e seus vários protocolos (https://www.ietf.org/rfc/rfc3986.txt):

```
ftp://ftp.is.co.za/rfc/rfc1808.txt

http://www.ietf.org/rfc/rfc2396.txt

ldap://[2001:db8::7]/c=GB?objectClass?one

mailto:contato@tableless.com.br

news:comp.infosystems.www.servers.unix

tel:+1-816-555-1212

telnet://192.0.2.16:80/

urn:oasis:names:specification:docbook:dtd:xml:4.1.2
```

Uma URI também pode ser definida em formato de URN (*Unified Resource Name*). Este funciona como se fosse um nome único para uma fonte de informação, como por exemplo o ISBN, para identificar os livros. Esse formato de URN só serve para identificação de um recurso, e não localização.

A Web Semântica não existe se não pudermos identificar as fontes de conteúdo. Se não há identificação, não há conexão e o relacionamento entre os recursos não existe. Então, como vou relacionar algo que não há nome? Ou como vou relacionar algo que eu não posso localizar?

Por este motivo as URIs e, principalmente, as URLs são importantes. Elas já fazem muito bem a tarefa de localizar e identificar, além de a internet ser baseada nesse padrão.

Agora que conseguimos interpretar, identificar e localizar as nossas fontes de informação, nós precisamos definir um contexto para que as máquinas possam conversar e se entender. Esse contexto é definido por **vocabulários**, que contêm os termos exatos usados nos diálogos entre as máquinas e que definirão as informações encontradas no seu site.

5.4 Para ler mais

- O que é Unicode? — http://www.unicode.org/standard/translations/portuguese.html
- Semantic Web Architecture — http://www.obitko.com/tutorials/ontologies-semantic-web/semantic-web-architecture.html

- BERNERS-LEE, T. FIELDING, R.; MASINTER, L. *Uniform Resource Identifier (URI):* Generic Syntax. 2005. Disponível em: https://tools.ietf.org/html/rfc3986.

- MOLINARI, Willian. *Desconstruindo a Web:* As tecnologias por trás de uma requisição. São Paulo: Casa do Código, 2016.

Capítulo 6
Vocabulários

6.1 Categorização e Aristóteles

Um vocabulário (ou uma ontologia) é um grupo de classificação de conceitos e relações. **Classificar é separar coisas em classes e grupos, usando uma forma de categorização**. Os seres humanos são bons em categorizar coisas e isso vem desde a Grécia antiga, com filósofos como Aristóteles e seu professor Platão.

Categorizar é simplesmente agrupar coisas de acordo com as suas características semelhantes. Sempre quando vemos algo, estamos categorizando esse "algo" de acordo com suas características em um grupo. Nossos cérebros fazem isso sem que percebamos na maioria das vezes. A categorização acontece para que não nos percamos dentro de contextos e também em ambiguidades.

Aristóteles foi um filósofo que desbravou e descreveu toda a complexidade envolvida na tarefa de categorizar as coisas do mundo. Em seu livro chamado **Categorias**, ele tenta explicar as 10 categorias que classificam tudo o que existe de conhecido no mundo. São elas:

1. **Substância**: homem, cachorro, cavalo;
2. **Quantidade**: três metros, oito quilômetros;
3. **Relação**: metade, maior, menor;
4. **Qualidade**: branco, preto, quadrado;
5. **Quando**: antes, depois, ontem;
6. **Onde**: em Paris, aqui;
7. **Estar-em-uma-posição**: está deitado, está sentado;
8. **Ter**: está armado, está calçado;
9. **Fazer**: cortar, queimar, amassar;
10. **Sofrer**: ser cortado, ser queimado.

Praticamente todas as últimas nove são predicados de Substância. **Entende-se por predicado aquilo que se declara sobre o sujeito. A Substância se entende como sujeito da oração**. Cada uma dessas coisas ditas sem nenhuma ligação significa, ou substância, ou quantidade, ou relação, ou qualidade, ou quando, ou onde, ou estar-em-uma-posição, ou ter, ou fazer, ou sofrer.

Todas essas coisas ditas de forma isolada são inúteis e não significam nada. Mas da união entre elas é que ocorre a afirmação. Por exemplo: homem *(substância/sujeito)* corre *(fazer)*.

Aqui já vimos o básico para entendermos o conceito de **triplas**, em que uma tripla é formada por um sujeito, ligado por um predicado ao objeto, que é praticamente o que Aristóteles explica em seu texto - claro que em um nível muito mais filosófico, evocando a qualidade da matéria, dos seres e de outras características do mundo.

Quando eu digo que alguém é branco, significa, que alguém tem a “qualidade” da brancura. O branco está em alguma coisa, nesse caso o sujeito, que é o homem. Esse branco não faz parte do homem, mas não pode estar fora daquilo em que está. Logo, essa brancura está em um corpo, porque todas as cores estão em algum lugar, mas não fazem parte essencial desse sujeito. A cor é uma coisa, mas onde essa cor está é outra coisa diferente.

Dentre outras tantas complexidades, Aristóteles começa a classificar tudo o que está no mundo, escolhendo determinados “vocabulários” para contextualizar nossas conversas e pensamentos. Sem essa categorização que fazemos naturalmente, que Aristóteles ajudou a expressar em seus estudos, não conseguiríamos distinguir diferentes sujeitos e seus predicados. Estaríamos perdidos em infinitas ambiguidades.

Não quero me aprofundar mais nessa parte filosófica da coisa, porque com certeza não sou a melhor pessoa para explicar a teoria aristotélica. Mas sugiro que você leia e pesquise sobre o texto **Categorias** de Aristóteles.

6.2 A importância de definir um vocabulário

Quando se está conversando com alguém ou com um grupo de pessoas, é comum que nesse diálogo haja um contexto. Vocês estão conversando sobre alguma coisa. Essa coisa pode ser uma pessoa, um objeto, um lugar, um evento, um número, um animal etc. Desde seu nascimento, você consegue classificar como cada uma dessas coisas faz parte de contextos diferentes.

Ora, você não conversa com alguém sobre o cruzamento de um país com o outro, pelo simples motivo de que “lugares” não cruzam, mas sim animais. Logo, quando você está falando sobre animais, você sabe exatamente em qual contexto a palavra “cruzamento” se encaixa e em quais ela não faz sentido. O uso dos vocabulários também evita alguns problemas de ambiguidade de termos utilizados. Quando falamos sobre **Paris**, podemos nos referir ao lugar, que fica na França, ou à modelo Paris Hilton.

Como disse, para os seres humanos, fazer essas distinções e entender o contexto de um diálogo é quase que natural, mas para as máquinas entender esse tipo de relação é impossível até que alguém determine sobre qual contexto essas informações fazem parte. Nós fazemos isso quando dizemos do que se trata cada fonte de informação, indicando de alguma forma (dependendo do modelo de dados que você está usando: RDFa, JSON-LD, Microdata etc.) qual vocabulário as duas máquinas devem usar para se referir àquela determinada fonte de informação.

Um vocabulário é um conjunto de propriedades de classificações, que serve para que as máquinas entendam qual a relação entre uma fonte de informação e outra, e quais as suas características individuais. Por exemplo: se estamos falando sobre uma empresa, existem termos comuns que usaremos durante nossa conversa, como o endereço da empresa, a pessoa que fundou, seus funcionários e departamentos etc. Todos esses termos devem estar mapeados em um vocabulário para que possamos utilizá-lo ao criar,

por exemplo, o site dessa empresa, de forma que uma máquina (o Google, por exemplo) consiga saber dessas informações consumindo essa página.

> A diferença entre os termos **VOCABULÁRIO** e **ontologia** não é clara. Não há uma separação definitiva entre os dois conceitos, por isso, é normal que você se confunda com essas duas palavras em suas pesquisas. Para simplificar, trate-os como a mesma coisa. No livro, vou usar o termo **vocabulário** para facilitar o entendimento. Mas há quem ache que **ontologia** é algo mais abstrato, estando mais relacionado à filosofia.

Como existe uma série de relações que podemos fazer, vamos precisar de uma série de diferentes vocabulários para cobrir todos os gêneros de assuntos ou, pelo menos, os mais comuns. Além de ter vocabulários para categorizarmos e contextualizarmos nossas informações, temos a necessidade de relacionar informações por meio de fontes de dados estruturados. Por exemplo: se estamos falando sobre uma pessoa, como o John Lennon, você precisará usar um vocabulário que tenha propriedades próprias para se referir sobre essa pessoa e também vai precisar indicar uma fonte de informações sobre ela.

Quais vocabulários podemos usar?

Nós indicamos os vocabulários por meio de uma URI. Nessa URI, são encontradas referências às propriedades que cada um dos vocabulários utiliza para formar um contexto. Geralmente, para localizar uma propriedade em um vocabulário específico, basta juntar o nome da classe ou propriedade na URL do vocabulário.

Como exemplo, podemos usar o Schema.org. Para visualizarmos o type `Person`, basta juntar a URI do vocabulário `http://schema.org/`, e colocar o nome da type logo depois: `http://schema.org/Person`. Nessa

página, você consegue saber todas as propriedades que são usadas na type `Person`.

Existem uma série de vocabulários disponíveis que podemos usar hoje em dia. Veja os mais populares e indicados pelo W3C:

- Schema.org (http://schema.org/) é uma iniciativa da comunidade, iniciada e patrocinada pelo Google, Yahoo!, Microsoft e Yandex, para criar vocabulários.
- FOAF, ou Fried-of-a-Fried (http://www.foaf-project.org/), é um conjunto de termos que descrevem a relação entre pessoas.
- SIOC (http://sioc-project.org/) é uma ontologia de termos que podem ser usados para descrever comunidades online.
- DOAP (https://github.com/edumbill/doap/wiki) é o projeto de um vocabulário para descrever projetos de software, principalmente projetos open source.
- Music Ontology (http://musicontology.com/) é um vocabulário para descrever coisas do meio musical como artistas, álbuns, músicas, arranjos etc.
- Organization Ontology (http://epimorphics.com/public/vocabulary/org.html) é um vocabulário que descreve uma estrutura organizacional.
- GoodRelations (http://www.heppnetz.de/ontologies/goodrelations/v1) é um vocabulário para descrever produtos, preços, lojas etc.

Se você procurar no Google, pode achar uma série de outros vocabulários. Você pode também pesquisar nos sites a seguir, que colecionam esses vocabulários. O W3C sugere que você tente reusar qualquer um desses antes de criar um novo.

- Swoogle - http://swoogle.umbc.edu/
- Watson - http://watson.kmi.open.ac.uk/WatsonWUI/
- Falcons - http://ws.nju.edu.cn/falcons/objectsearch/index.jsp

Veja um exemplo de uso em JSON-LD do vocabulário Schema, do tipo `Person`:

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "name": "Diego Eis"
}
```

Agora, veja a mesma coisa só que usando Microdata:

```
<div itemtype="http://schema.org/Person" itemscope>
  <span itemprop="name">Diego Eis</span>
</div>
```

E a mesma coisa em formato RDFa:

```
<div xmlns="http://www.w3.org/1999/xhtml"
  prefix="
    schema: http://schema.org/
    rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    rdfa: http://www.w3.org/ns/rdfa#
    rdfs: http://www.w3.org/2000/01/rdf-schema#"
  >
    <div typeof="schema:Person" property="schema:name">Diego Eis</div>
</div>
```

6.3 Para ler mais

- Guia de Web Semântica - Vocabulários e Ontologias — http://ceweb.br/guias/web-semantica/capitulo-6/
- Livro *The Categories* de Aristóteles, disponível em domínio público — http://www.dominiopublico.gov.br/pesquisa/DetalheObraForm.do?select_action=&co_obra=13726
- Introducing RDFS & OWL — http://www.linkeddatatools.com/introducing-rdfs-owl
- Categorização — https://pt.wikipedia.org/wiki/Categorização

- Common Vocabularies / Ontologies / Micromodels, em W3C — https://www.w3.org/wiki/TaskForces/CommunityProjects/LinkingOpenData/CommonVocabularies
- BRICKLEY, Dan; MILLER, Libby. *The Semantic Web:* Introduction. 2014. Disponível em: http://xmlns.com/foaf/spec/#sec-sw.
- MORICI, Igor Mota. *As categorias de Aristóteles e suas categorias.* 2008. Disponível em: http://www.bibliotecadigital.ufmg.br/dspace/bitstream/handle/1843/BUBD-9FUHJC/immorici_categorias.pdf?sequence=1.
- PICKLER, Maria Elisa Valentim. *Web Semântica:* ontologias como ferramentas de representação do conhecimento. Perspectivas em Ciência da Informação, v. 12, n. 1, Belo Horizonte, Jan./Abr. 2007. Disponível em: http://www.scielo.br/scielo.php?script=sci_arttext&pid=S1413-99362007000100006.

Capítulo 7
Triplas e grafos

> *We live in a society bloated with data, yet starved for wisdom. We're connected 24/7 , yet anxiety, fear, depression and loneliness is at an all-time high. We must course-correct.* —
> **Elizabeth Kapu'uwailani Lindsey**

Como as máquinas conversam na Web? Atualmente, é muito difícil que um sistema não precise consumir dados de outros sistemas. Mesmo quando fazemos um único sistema, é comum o quebrarmos em diversas aplicações menores, com o objetivo de facilitar a manutenção e separar as responsabilidades. Criamos uma cultura de microsserviços e fazemos com que todos esses sistemas se conversem, resultando em um produto, site etc. que as pessoas usarão.

Basicamente essa mesma ideia é aplicada quando queremos que determinado sistema consuma dados de um sistema de terceiros. Imagine que seu site/produto precise consumir dados do GitHub. Provavelmente, você criará uma integração do seu sistema, de forma que ele consuma a API do GitHub.

É o mesmo tipo de conversa que falamos anteriormente. Nesse nível, podemos falar com uma API feita em SOAP, que responde XML, ou feita em Rest, na qual temos uma resposta mais flexível para outros formatos. Seu sistema pede o dado, o outro sistema responde por um *endpoint*, e pronto. Esses dados vão e vêm, assim nós conseguimos fazer com que a internet “converse”.

Aliás, a internet é baseada em um tipo de conversa também. Quando falamos de protocolo HTTP, falamos sobre conversa entre cliente e servidor. O cliente pede um bocado de dados, o servidor responde. Existem dois tipos de mensagens: os **requests**, que são

enviados pelo cliente pedindo alguma coisa para o servidor; e as **responses**, que são as respostas que o servidor fornece.

Tanto os dados trocados entre os sistemas quanto as mensagens trocadas pelo cliente e servidor têm estruturas específicas. Essas estruturas são previamente combinadas para que as duas pontas interpretem corretamente seus dados. Quando falamos sobre Web Semântica, falamos sobre a disponibilização de dados para qualquer um. Logo, esses dados devem estar em uma estrutura padrão, que facilite o relacionamento e o consumo dos dados.

7.1 Como estruturamos dados hoje

Existem algumas estruturas de dados comuns que usamos todos os dias e, por isso, estamos bem acostumados com elas. Um exemplo bem simples é a utilização de dados tabulares. Usar uma planilha é algo tão trivial e fácil de entender que qualquer um consegue aprender de maneira rápida:

Restaurante	Endereço	Especialidade	Preço	Aber
Vento Haragano	Av. Rebouças, 1001	Churrascaria	$$$$	Terça-Domin (10:00 22:00)
Aoyama	R. Dr. Fonseca Brasil, 108	Japonesa	$$$	Segun Domin (11:00 22:00)
Família Mancini	R. Avanhandava, 81	Italiana	$$$	Segun Domin (11:30 00:00)

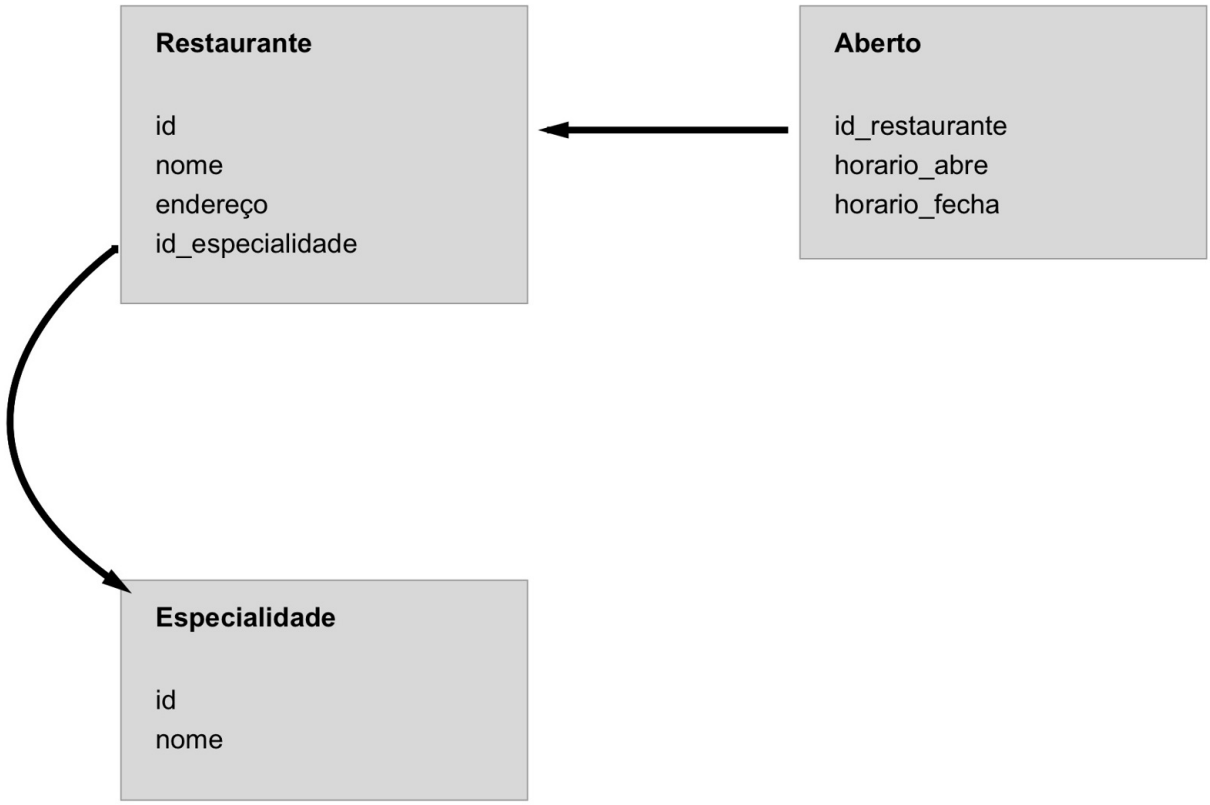
Não existe muito segredo nessa estrutura de dados, ela é facilmente entendida por seres humanos. Você consegue saber o significado da célula pelo nome da sua coluna. Você sabe que **Vento Haragano** se trata de um **Restaurante**, e sua **Especialidade** é **Churrasco** apenas cruzando o valor da célula com o nome da coluna.

Mas no caso da tabela, se precisarmos manipular melhor esses dados, extraindo ou guardando informações mais úteis, eventualmente uma única célula guardaria mais de um dado. Como acontece na coluna **Aberto**, que além de guardar o dia da semana, guarda também o horário de funcionamento do estabelecimento.

> Estou usando um exemplo bem bacana que um cara chamado Luis Cipriani fez em sua palestra (https://www.youtube.com/watch?v=Ppa98EPPssE) sobre Web Semântica, lá em 2014 em um dos evento do GuruSP na Abril.

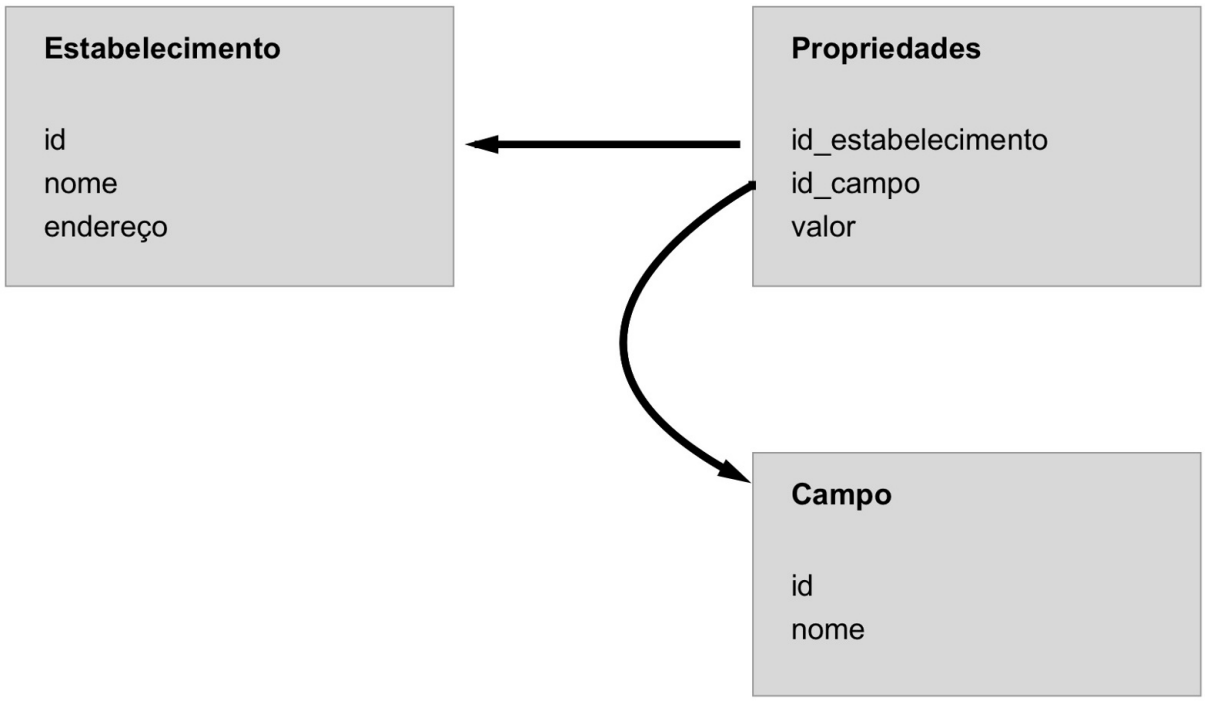
Obviamente guardar grande quantidade de dados em formato tabular não é a melhor saída. Ainda bem que existem os bancos de dados relacionais.

Bancos de dados relacionais como PostgreSQL e MySQL são bem comuns no mercado, e seu funcionamento é bem conhecido por todos os programadores, principalmente os back-end. A ideia de um banco de dados relacional é que você relacione uma grande quantidade de dados, de forma organizada.



O problema disso é que, se você não pensou direito na estrutura do banco, de forma que essa estrutura seja genérica o suficiente para sofrer mudanças futuras, em algum momento você será obrigado a mudar o schema do banco, e isso é um problema. Você pode querer não se referir mais a restaurantes, mas a qualquer tipo de estabelecimento.

Não vou entrar no problema que é mudar um schema de banco de dados. Mas para tentar se prevenir desse problema, os devs adotam uma estratégia de criar um schema flexível, sendo possível atribuir várias propriedades para um estabelecimento de acordo com as suas necessidades:



Uma representação em tabela ficaria algo mais ou menos assim:

estabelecimento

id	nome	endereço
1	Vento Haragano	Av. Rebouças, 1001
2	Aoyama	R. Dr. Fonseca Brasil, 108
3	Família Mancini	R. Avanhandava, 81

propriedades

id_estabelecimento	id_campo	valor
1	1	churrasco
2	2	$$
3	1	italiana

campo

campo	nome
1	especialidade
2	preço
3	...

Para facilitar, podemos tirar a tabela de **campo**, e colocar o nome do campo diretamente na coluna `campo`. Nesse caso, teríamos de tomar cuidado com a consistência dos nomes dos campos. Ficaria assim:

estabelecimento

id	nome	endereço
1	Vento Haragano	Av. Rebouças, 1001
2	Aoyama	R. Dr. Fonseca Brasil, 108
3	Família Mancini	R. Avanhandava, 81

propriedades

id_estabelecimento	id_campo	valor
1	especialidade	churrasco
2	preço	$$
3	especialidade	italiana

Okay, okay. Sei que você é um cara muito melhor do que eu, e talvez não faria dessa forma. Mas meu objetivo aqui é só ilustrar como chegaríamos em um formato de estrutura de dados que, na Web Semântica, chamamos de triplas.

Perceba ali na tabela de **Propriedades**, que nós temos o código do Restaurante (imagine que poderíamos colocar ali o nome do Restaurante, e não seu código). Nós temos o nome da sua especialidade, preço etc. e, logo após, o valor da propriedade. Em uma tabela que representa triplas, isto ficaria assim:

triplas

sujeito	predicado	objeto
Vento Haragano	especialidade	churrasco
Aoyama	preço	$$
Vento Haragano	preço	$$$$
Aoyama	especialidade	japonesa
Família Mancini	preço	$$$
…	…	…

O formato `sujeito > predicado > objeto`, conhecido como triplas (do inglês *triples*), é o formato base da organização, estruturação e relacionamento de dados na Web Semântica. É o que veremos a seguir.

7.2 Triplas: a base da estrutura de dados na Web Semântica

As triplas são o formato mais básico que temos para relacionar e estruturar dados na Web Semântica. É toda a base que você precisa saber sobre o que vamos discutir daqui para a frente.

Um modelo é um conjunto de declarações. Uma declaração é formada pelo **sujeito**, **predicado** e **objeto**. O predicado e o sujeito

são recursos. O objeto pode ser um recurso, ou um literal. O formato dessas declarações é bem simples e sempre segue a estrutura:

SUJEITO > PREDICADO > OBJETO

A vantagem desse formato simples é que podemos extrair dela um **grafo**:

Figura 7.6: Estrutura da tripla

Essa declaração expressa a relação entre duas fontes. O **sujeito** e o **objeto** representam as duas fontes que serão relacionadas pelo **predicado**, que representa a natureza dessa relação. A relação é formulada na direção do sujeito para o predicado, que no RDF é chamada de **propriedade**.

Imagine o RDF como sendo uma linguagem para expressar **declarações** acerca dos recursos, facilitando a comunicação entre máquinas. Para ficar mais fácil de entender as triplas, veja algumas expressões a seguir:

1. **Diego** > *é uma* > **pessoa**.
2. **Diego** > *é casado com* > **Marcela**.
3. **Diego** > *nasceu em* > **03 de Dezembro de 1983**.
4. **Diego** > *é interessado na* > **Amazon**.
5. **Amazon** > *foi criada pelo* > **Jeff Bezos**.
6. **O livro A Loja De Tudo** > *é sobre a* > **Amazon**.

A mesma fonte é referenciada em múltiplas **triplas**. Nesses exemplos, **Diego** é o assunto em quatro **triplas**, e a **Amazon** é o assunto de um trio (5) e o objeto de dois (4 e 6). Essa possibilidade de ter a mesma fonte como sujeito e objeto em várias triplas é o que possibilita encontrar conexões entre elas.

> **Muita atenção agora**
>
> **Tudo** o que veremos a seguir e nos próximos capítulos tem a ver, de alguma maneira, com as triplas. Quando falarmos sobre RDF, RDFa, JSON-LD, Microformats, Turtle etc., estamos apenas estudando formas de representar triplas por meio de códigos que podem ser consultados posteriormente. Quando falarmos sobre Linked Data, estamos falando sobre disponibilizar nossos dados (escritos usando alguma linguagem de representação de triplas) na Web.

Vamos começar entendendo como podemos representar as triplas de forma visual, os famosos **grafos**.

7.3 Grafos: representando triplas visualmente

As triplas podem ser representadas em forma de grafos, que é um formato muito mais fácil de enxergar os relacionamentos entre as informações. Os grafos consistem em **nós** e **setas**. Os **sujeitos** e **objetos** são os **nós** no gráfico. Os **predicados** são as **setas**.

Se fôssemos fazer um gráfico das **triplas** do exemplo anterior, ficaria mais ou menos assim:

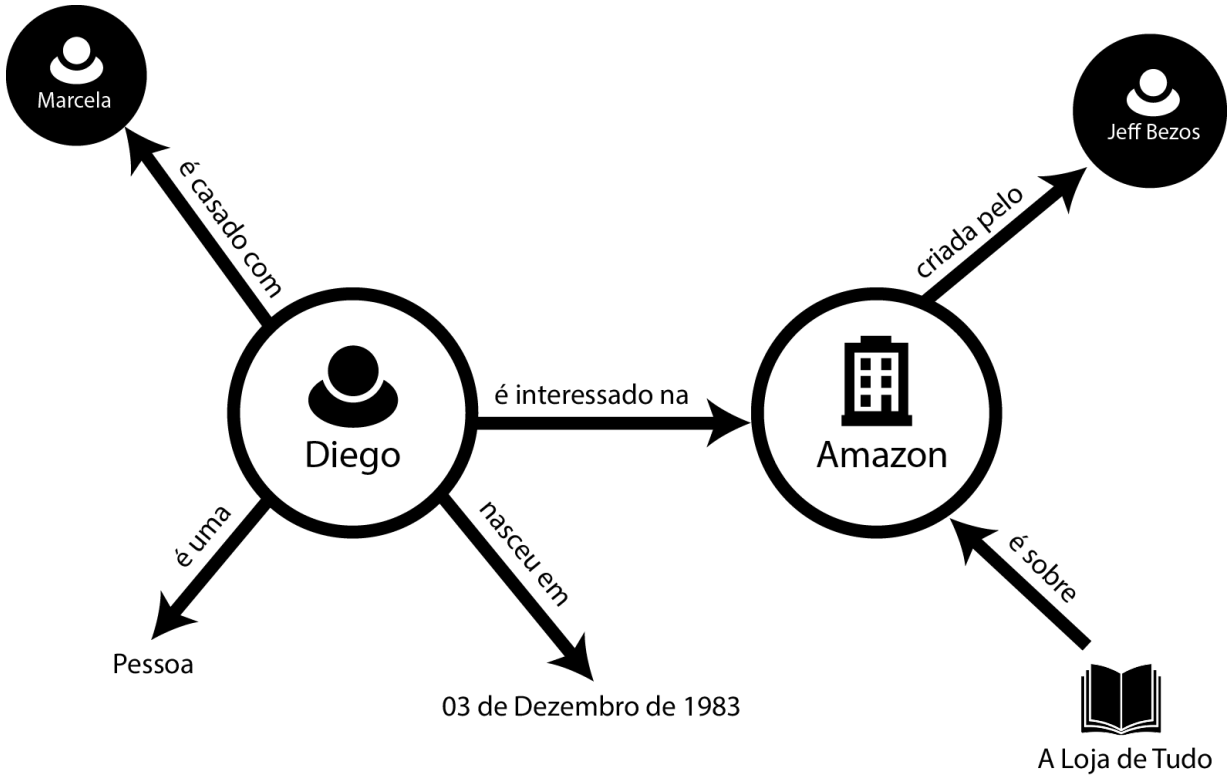


Figura 7.7: Como um grafo pode ser visualizado

Essa é a melhor forma de ver a relação entre as informações, exatamente porque conseguimos visualizar a densidade de relações entre cada fonte de informação. O Google já usava essa analogia para explicar como o PageRank funciona.

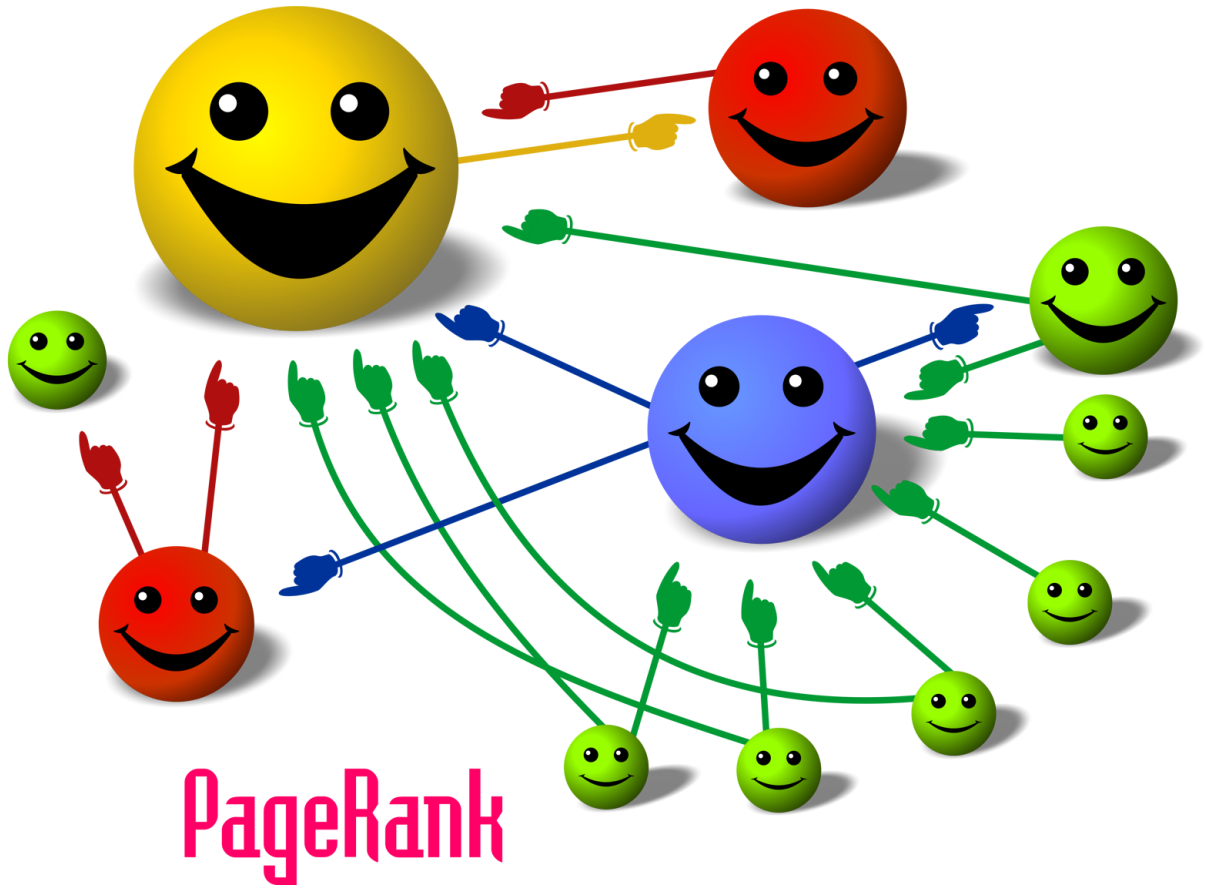


Figura 7.8: Fonte: https://pt.wikipedia.org/wiki/PageRank#/media/File:PageRank-hi-res.png

7.4 Representando as triplas com código

Nós conseguimos representar as triplas usando RDF, JSON-LD, Microdata e outras linguagens. O interessante é que você marcará seu HTML ou disponibilizará seus dados usando JSON-LD, que representará as triplas e que poderão ser exportadas em formato de grafos.

Assim, as triplas estarão representadas no seu código, marcando os dados encontrados. Veja como exemplo o seguinte texto:

```
Olá, eu me chamo Diego Eis, nasci no dia 03/12/1983 e atualmente trabalho
como Gerente de Produtos na Easynvest, que foi fundada no ano de 1968 .
```

```
Você pode entrar em contato comigo no e-mail diego@tableless.com.br e ler
alguns textos que escrevo sobre Gestão e Desenvolvimento Web no meu blog
pessoal.
```

Dentro desse texto, existe uma série de informações importantes sobre mim. Você consegue descobrir quando nasci, meu nome, meu e-mail, onde trabalho etc. Vamos marcar esse texto usando RDFa, destacando as informações importantes:

```
<div vocab="http://schema.org/" typeof="Person">
  Olá, eu me chamo
  <a property="image"
href="https://secure.gravatar.com/avatar/1bf877955dc2e43662320fd3b0280166?
size=800">
    <span property="name">Diego Eis</span>
  </a>,
  nasci no dia <span property="birthDate">03/12/1983</span> e atualmente
trabalho como
  <span property="jobTitle">Coordenador</span> na
  <span property="worksFor" typeOf="Organization">
    <span property="legalName">Locaweb</span>, que foi fundada no ano de
<span property="foundingDate">1998</span>
  </span>.
  Você pode entrar em contato comigo no e-mail <a property="email"
href="mailto:diego@tableless.com.br">diego@tableless.com.br</a> e ler
alguns textos que escrevo sobre Gestão e Desenvolvimento Web no meu <a
property="url" href="http://diegoeis.com/">blog pessoal</a> .
</div>
```

Com esses dados marcados, conseguimos extrair as triplas. Usando o RDFa/Play (um sistema que transforma código RDFa em grafos), você tem uma representação das entidades encontradas no texto:

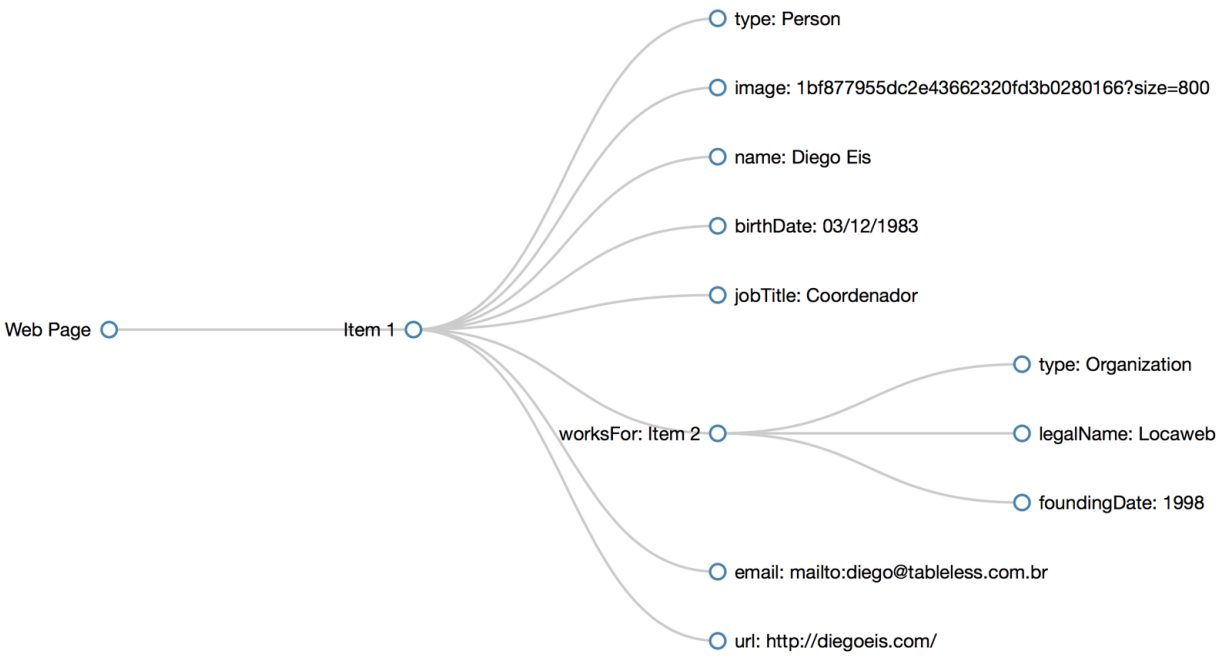
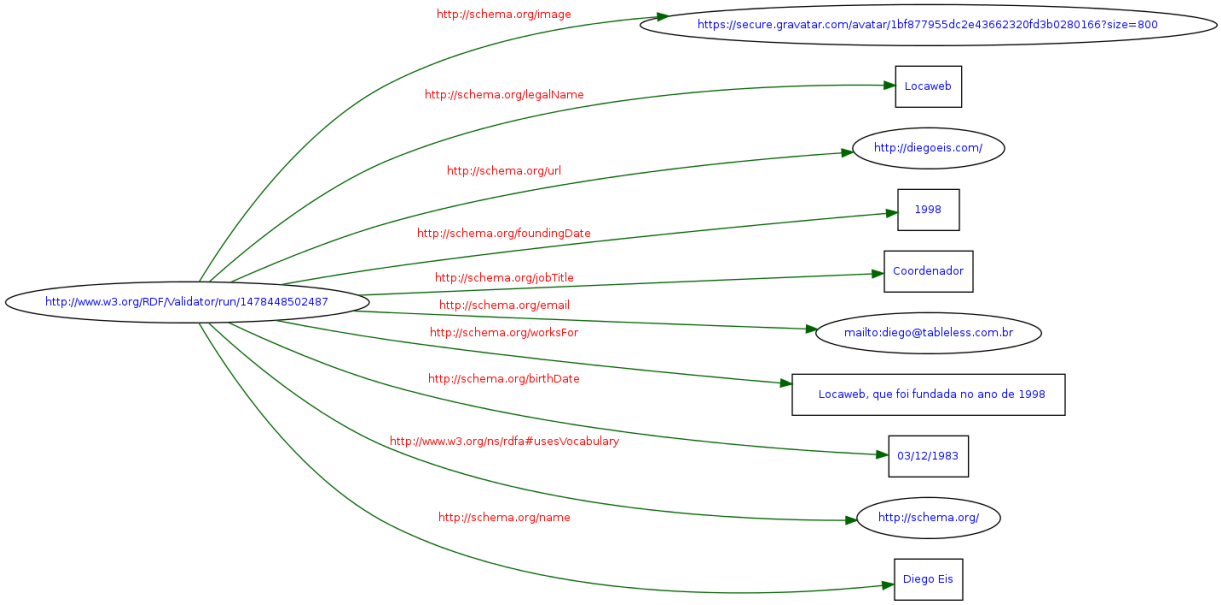


Figura 7.9: Propriedades encontradas no texto

Com essas triplas encontradas, conseguimos montar um grafo mais ou menos assim:



Além de poder marcar os dados no HTML, nós podemos usar outros formatos, que são mais fáceis de manipular via linguagens de query. Os formatos variam bastante, e o formato a seguir é chamado de Turtle:

```
@prefix rdfa: <http://www.w3.org/ns/rdfa#> .
@prefix schema: <http://schema.org/> .

<http://diegoeis.com>
  rdfa:usesVocabulary schema: ;
  a schema:Person ;
  schema:image
<https://secure.gravatar.com/avatar/1bf877955dc2e43662320fd3b0280166?
size=800> ;
  schema:name "Diego Eis" ;
  schema:birthDate "03/12/1983" ;
  schema:jobTitle "Coordenador" ;
  schema:worksFor """
    Locaweb, que foi fundada no ano de 1998
  """ ;
  schema:legalName "Locaweb" ;
  schema:foundingDate "1998" ;
  schema:email <mailto:diego@tableless.com.br> ;
  schema:url <http://diegoeis.com/> .
```

Um outro formato, chamado N-Triples, tem uma estrutura que se parece mesmo com uma tripla escrita, assim:

```
<http://diegoeis.com> <http://www.w3.org/ns/rdfa#usesVocabulary>
<http://schema.org/> .
<http://diegoeis.com> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
<http://schema.org/Person> .
<http://diegoeis.com> <http://schema.org/image>
<https://secure.gravatar.com/avatar/1bf877955dc2e43662320fd3b0280166?
size=800> .
<http://diegoeis.com> <http://schema.org/name> "Diego Eis" .
<http://diegoeis.com> <http://schema.org/birthDate> "03/12/1983" .
<http://diegoeis.com> <http://schema.org/jobTitle> "Coordenador" .
<http://diegoeis.com> <http://schema.org/worksFor> "\n    Locaweb, que foi
fundada no ano de 1998\n  " .
<http://diegoeis.com> <http://schema.org/legalName> "Locaweb" .
<http://diegoeis.com> <http://schema.org/foundingDate> "1998" .
<http://diegoeis.com> <http://schema.org/email>
<mailto:diego@tableless.com.br> .
<http://diegoeis.com> <http://schema.org/url> <http://diegoeis.com/> .
```

Esses formatos são usados pelas *Triple Stores* para guardarem seus dados. As Triple Stores são serviços online cuja função é simplesmente guardar dados em forma de triplas. Elas distribuem esses dados para que você consiga relacioná-los ou extrair informação.

7.5 Triple Stores — Bases de conhecimento

Nós procuramos informações em muitos lugares. A Wikipédia é sem dúvida uma das fontes de informação mais conhecidas da internet. Imagine se todas as informações da Wikipédia estivessem disponíveis em um banco de dados público, que pode ser consultado e reutilizado por você. É o que a DBPedia (http://dbpedia.org/) faz.

A DBPedia é uma base de conhecimento mantida pela comunidade, que extrai as informações da Wikipédia e da Wikidata, disponibilizando isso em formato comum, semântico, e usando todos os conceitos da Linked Data. Ela oferece acesso via URIs, HTTP, HTML, RDF e SPARQL.

Se você quiser, por exemplo, relacionar um artigo sobre o John Lennon, você pode relacionar com essa página (http://dbpedia.org/page/John_Lennon), onde há informações sobre ele. Ou se você estiver falando sobre o Monte Everest (http://dbpedia.org/page/Mount_Everest), é essa página que você deve relacionar.

Como a DBPedia é um conjunto de dados extraídos da Wikipédia, para encontrar um recurso da DBPedia, você usa a mesma estrutura de dados: na Wikipédia, a URL do John Lennon seria **https://en.wikipedia.org/wiki/John_Lennon**. Na DBPedia, a URL seria **http://dbpedia.org/page/John_Lennon**.

Veja que o formato é *https://en.wikipedia.org/wiki/**nome**** e *http://dbpedia.org/page/**nome****.

Os Datasets são importantes porque publicam dados de domínio público. Se não fosse assim, você precisaria fazer um scrap de dados, ou ser o Google, para explorar, minerar, tratar, estruturar e, só depois, usar esses dados. No capítulo *Linked Data: o conceito de dados conectados*, você já viu imagem a seguir, mas queria mostrar agora como está o status do relacionamento de dados na internet usando esses datasets:

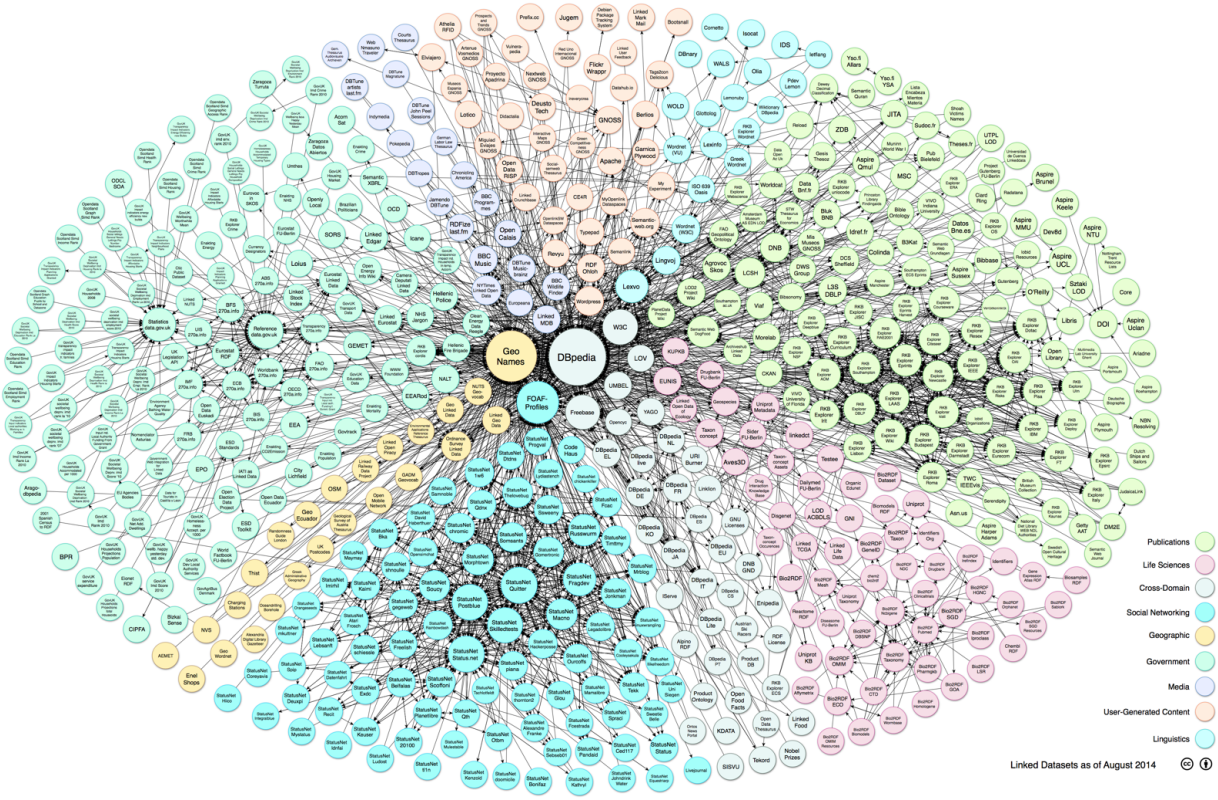


Figura 7.11: Fonte: http://lod-cloud.net/versions/2014-08-30/lod-cloud_colored.png

Existe uma série de datasets disponíveis por aí. Praticamente todos eles disponíveis para uso online, ou para fazer downloads. Veja alguns deles a seguir:

- dados.gov.br — http://dados.gov.br
- WikiData — https://www.wikidata.org/wiki/Wikidata:Main_Page

- DBPedia — http://dbpedia.org
- AWS Public Datasets — https://aws.amazon.com/datasets/
- data.gov — http://catalog.data.gov/dataset
- Unicef Data — https://data.unicef.org
- Freebase — http://www.freebase.com/

Tendo em mente toda essa introdução sobre como a organização da informação fundamentou a internet que conhecemos hoje, podemos falar sobre como a internet está evoluindo para construir um ambiente mais colaborativo entre humanos e máquinas. E faz isso usando toda a informação disponível na Web, de forma organizada, estruturada e, principalmente, conectada entre si. Logo, o que de fato é a Web Semântica?

7.6 Concluindo

Já sabemos o que são triplas e como nossas informações serão representadas por meio de grafos. Mas como aplicamos isso **hoje**, em sistemas e sites que já temos desenvolvidos?

Nos próximos capítulos, vamos ler sobre como fazer isso usando RDFa e JSON-LD.

7.7 Para ler mais

- Structured Data Linter — http://linter.structured-data.org
- Semantic Community - Graph Database Presentation — http://semanticommunity.info/Data_Science/Graph_Databases#Slide_9_3._Data_Modeling_with_Graph
- RDFa 1.1 Distiller and Parser — https://www.w3.org/2012/pyRdfa/Overview.html
- RDFa/Play — https://rdfa.info/play/

- EasyRDF Converter — http://www.easyrdf.org/converter
- DBPedia — http://pt.dbpedia.org/page/
- LIBKIN, Leonid; REUTTER, Juan; VRGOC, Domagoj. *TriAL for RDF:* Adapting Graph Query Languages for RDF Data. 2013. Disponível em: http://jreutter.sitios.ing.uc.cl/pods13.pdf.
- SEQUEDA, Juan. *Introduction to Triplestores*. 2013. Disponível em: http://www.dataversity.net/introduction-to-triplestores/.

Capítulo 8
RDFa

8.1 Um pouco sobre RDF

Não podemos falar sobre RDFa sem falar antes sobre RDF. Praticamente todos os livros sobre Web Semântica da face da Terra incluem um capítulo inteiro sobre RDF. O RDF é um framework, mas não como os frameworks que conhecemos por aí. Ele é um framework de conceitos. Você não "programa" ou "codifica" em RDF, você usa o XML ou outras linguagens como JSON-LD para aplicar os conceitos que o RDF prega.

O RDF é o padrão do W3C para representação desses grafos. O tipo de estrutura de relacionamento do RDF traduz um punhado de código XML em gráficos, que é a melhor forma para visualizarmos os relacionamentos entre cada recurso de informação.

Estes são os elementos do modelo de dados do RDF:

- Recurso (*Resource*);
- Propriedade (*Properties*);
- Valor (*Value*);
- Declaração (*Statements*).

Para representar um grafo, os **nós** e os **predicados** são escritos usando as regras do XML, como nomes de elementos, atributos, valores de atributos e, finalmente, com os conteúdos dos elementos.

Como vimos em capítulos anteriores, um grafo é desenhado por uma coleção de caminhos, formados por nós que se conectam por meio de predicados (setas) com outros nós. Em RDF/XML, esse conceito se declara com elementos dentro de elementos, em que os elementos são os nós. A documentação do W3C dá um exemplo:

```
<?xml version="1.0"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
         xmlns:dc="http://purl.org/dc/elements/1.1/"
         xmlns:ex="http://example.org/stuff/1.0/">

  <rdf:Description rdf:about="http://www.w3.org/TR/rdf-syntax-grammar"
           dc:title="RDF/XML Syntax Specification (Revised)">

    <ex:editor>
      <rdf:Description ex:fullName="Dave Beckett">
            <ex:homePage rdf:resource="http://purl.org/net/dajobe/" />
      </rdf:Description>
    </ex:editor>

  </rdf:Description>

</rdf:RDF>
```

Esse código forma o seguinte grafo:

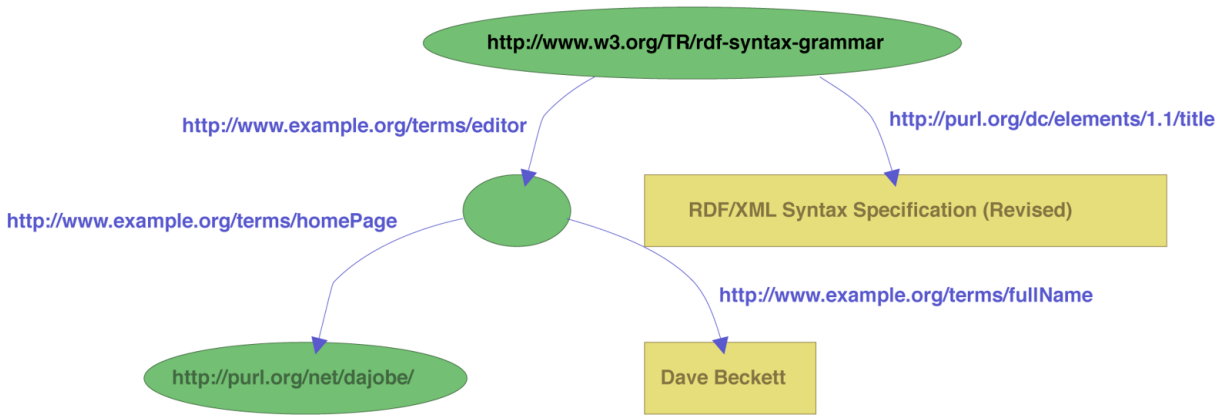


Figura 8.1: Grafo de exemplo do W3C

Não me aprofundarei muito no RDF, exatamente porque nos envolveríamos em um monte de besteiras que você nunca utilizará em seu dia a dia. Por isso, vamos ao que interessa: RDFa.

8.2 RDFa para salvar nossa sanidade

A web é feita por HTML, não por XML. Todo esse HTML guarda uma quantidade significativa de dados que serão explorados, mas, para isso, precisamos marcá-los de forma que os dados fiquem expostos. Para tanto, nós usamos o formato RDFa, que é uma serialização do RDF para ser incorporada no HTML por meio de atributos. Isso quer dizer que você não precisa de um arquivo XML para representar seu RDF.

O RDFa (*RDF in attributes*) é uma especificação de atributos para expressar dados estruturados em linguagens de marcação. Como só usamos HTML hoje, é nela que vamos nos focar.

O RDFa especifica a sintaxe dos atributos. Os termos de vocabulários não são sua responsabilidade, por isso, vamos usar o **Schema.org**, por ser bem popular.

O uso básico do RDFa é simples: você pega um pedaço do código que quer marcar, define o vocabulário e o tipo, depois marca as informações do HTML com os atributos. Veja um exemplo:

```
<p>
  Steven Paul Jobs, mais conhecido como Steve Jobs, foi um americano inventor, empresário e magnata americano no setor da informática. Ele foi fundador da Apple e várias outras empresas. Nasceu em São Francisco, no dia 24 de Fevereiro de 1955 e morreu em Palo Alto, no dia 5 de Outubro de 2011.
</p>
```

Agora, rechearemos esse pedaço de texto com atributos RDFa, usando o vocabulário de `Person` do Schema.org:

```
<p vocab="http://schema.org" typeof="Person">
  <span property="name">Steven Paul Jobs</span>, mais conhecido como <span property="alternateName">Steve Jobs</span>, casado com <span property="spouse">Laurene Powell Jobs</span>, foi um americano inventor, empresário e magnata americano no setor da informática. Ele foi fundador da <span property="owns">Apple</span> e várias outras empresas. Nasceu em <span property="birthPlace">São Francisco</span>, no dia <span property="birthDate">24 de Fevereiro de 1955</span> e morreu em <span
```

```
property="deathPlace">Palo Alto</span>, no dia <span
property="deathDate">5 de Outubro de 2011</span>.
</p>
```

A nível de curiosidade, a mesma coisa escrita em RDF/XML puro ficaria mais ou menos assim:

```
<?xml version="1.0" encoding="UTF-8"?>
<rdf:RDF
   xmlns:ns1="http://http://schema.org"
   xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfa="http://www.w3.org/ns/rdfa#"
>
  <rdf:Description rdf:about="">
    <rdfa:usesVocabulary rdf:resource="http://schema.org"/>
    <ns1:schema.orgbirthDate>24 de Fevereiro de
1955</ns1:schema.orgbirthDate>
    <ns1:schema.orgalternateName>Steve Jobs</ns1:schema.orgalternateName>
    <ns1:schema.orgbirthPlace>São Francisco</ns1:schema.orgbirthPlace>
    <ns1:schema.orgowns>Apple</ns1:schema.orgowns>
    <ns1:schema.orgspouse>Laurene Powell Jobs</ns1:schema.orgspouse>
    <ns1:schema.orgname>Steven Paul Jobs</ns1:schema.orgname>
    <ns1:schema.orgdeathDate>5 de Outubro de
2011</ns1:schema.orgdeathDate>
    <ns1:schema.orgdeathPlace>Palo Alto</ns1:schema.orgdeathPlace>
  </rdf:Description>
</rdf:RDF>
```

Isso resultaria em uma tripla assim:

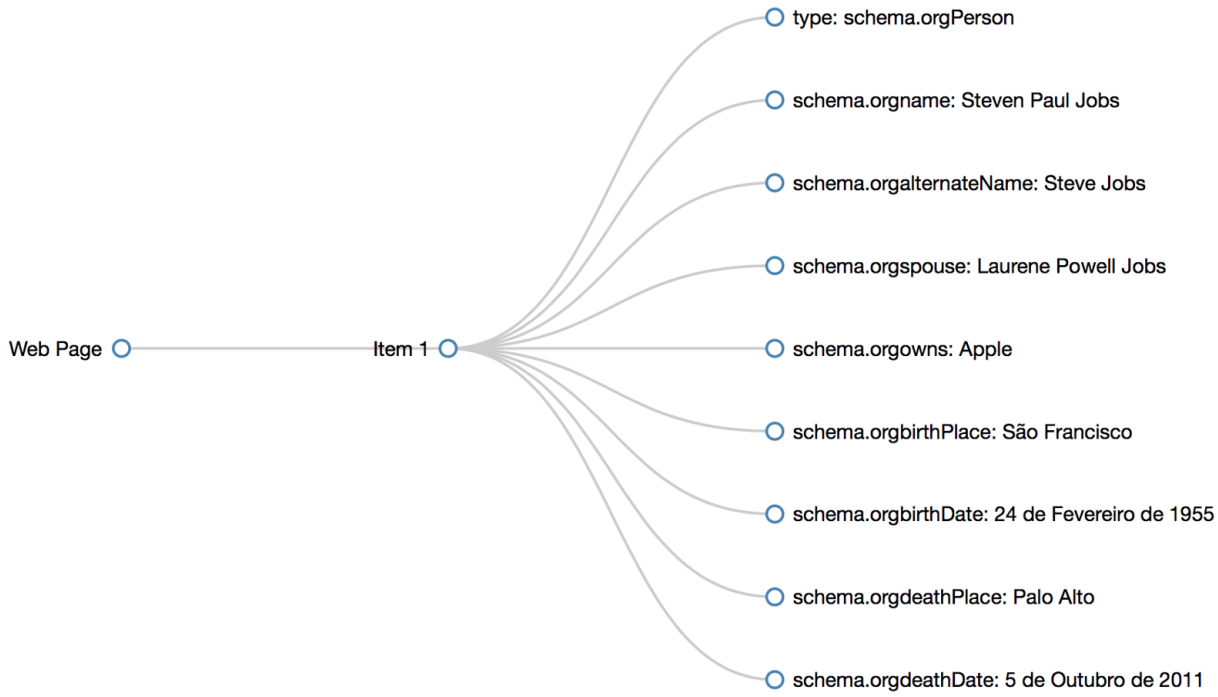


Figura 8.2: Utilizei o RDFa Play para transformar o código RDFa em triplas (https://rdfa.info/play/)

Para facilitar o trabalho, nós não precisamos definir o atributo `vocab` em cada pedaço que marcarmos com RDFa. O contexto será definido no início do HTML:

```
<html prefix="schema:http://schema.org/">
  <head>
    <title>RDFa</title>
  </head>
  <body>
  …
  </body>
</html>
```

Assim, nós usaremos no nosso HTML os atributos `typeof` para definir a qual categoria de **termos** estamos nos referindo. No caso do nosso exemplo, seria `Person`:

```
<html prefix="schema:http://schema.org/">
  <head>
    <title>RDFa</title>
```

```
  </head>
  <body>
     <p typeof="schema:Person"> … </p>
  </body>
</html>
```

Perceba que usei o prefixo `schema` antes de `Person`. O RDFa usa esses namespaces, porque podemos usar vários vocabulários em um mesmo documento se quisermos, bastando inserir outro endereço no atributo `prefix`:

```
<html prefix="schema:http://schema.org/
              foaf: http://xmlns.com/foaf/0.1/">
  <head>
    <title>RDFa</title>
  </head>
  <body>
     <p typeof="schema:Person"> … </p>
  </body>
</html>
```

8.3 Terminologia RDF

A estrutura de dados do RDFa retorna uma coleção de *declara ões*. Como já vimos, essas declarações são **triplas**, que significam uma unidade básica de informação que foi construída. No nosso exemplo, as triplas seriam assim:

```
Steven Paul Jobs tem um alternativo de Steve Jobs
Steven Paul Jobs é casado com Laurene Powell Jobs
Steven Paul Jobs era dono da Apple
Steven Paul Jobs nasceu em São Francisco
Steven Paul Jobs nasceu no dia 24 de Fevereiro de 1955
Steven Paul Jobs morreu em Palo Alto
Steven Paul Jobs morreu no dia 5 de Outubro de 2011
```

Como sabemos, podem haver um monte de **Steven Paul Jobs** pelo mundo. Por isso, pensando em uma das regras áureas da Web Semântica, vamos usar uma URL no lugar do termo **Steven Paul Jobs** para se referir ao sujeito. Estou usando aqui como referência o endereço do Steve Jobs do dataset da DBPedia:

```
<http://dbpedia.org/resource/Steve_Jobs> tem um alternativo de Steve Jobs
<http://dbpedia.org/resource/Steve_Jobs> é casado com Laurene Powell Jobs
<http://dbpedia.org/resource/Steve_Jobs> era dono da Apple
<http://dbpedia.org/resource/Steve_Jobs> nasceu em São Francisco
<http://dbpedia.org/resource/Steve_Jobs> nasceu no dia 24 de Fevereiro de
1955
<http://dbpedia.org/resource/Steve_Jobs> morreu em Palo Alto
<http://dbpedia.org/resource/Steve_Jobs> morreu no dia 5 de Outubro de
2011
```

Vamos usar também uma URL para definir os objetos, que são a terceira parte da tripla. Existem valores ali que não podem ser identificados com uma URL, e esses dados são definidos com aspas duplas, indicando que são literais (*Plain Literals*), uma string básica sem tipo ou informação de idioma. Fica mais ou menos assim:

```
<http://dbpedia.org/resource/Steve_Jobs>
    tem um alternativo de
    "Steve Jobs"

<http://dbpedia.org/resource/Steve_Jobs>
    é casado com
    <http://dbpedia.org/resource/Laurene_Powell_Jobs>

<http://dbpedia.org/resource/Steve_Jobs>
    era dono da
    <http://dbpedia.org/resource/Apple_Inc.>

<http://dbpedia.org/resource/Steve_Jobs>
    nasceu em
    <http://dbpedia.org/resource/San_Francisco>

<http://dbpedia.org/resource/Steve_Jobs>
    nasceu no dia
```

```
    "24 de Fevereiro de 1955"

<http://dbpedia.org/resource/Steve_Jobs>
    morreu em
    <http://dbpedia.org/resource/Palo_Alto,_California>

<http://dbpedia.org/resource/Steve_Jobs>
    morreu no
    dia "5 de Outubro de 2011"
```

Agora podemos identificar também uma URL para cada um dos **predicados** das nossas triplas. Os predicados indicam a relação entre o sujeito e o objeto. Em um grafo, elas seriam as setas que ligam uma fonte à outra. Fica assim:

```
<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/alternateName>
    "Steve Jobs"

<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/spouse>
    <http://dbpedia.org/resource/Laurene_Powell_Jobs>

<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/owns>
    <http://dbpedia.org/resource/Apple_Inc.>

<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/birthPlace>
    <http://dbpedia.org/resource/San_Francisco>

<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/birthDate>
    "24 de Fevereiro de 1955"

<http://dbpedia.org/resource/Steve_Jobs>
    <http://schema.org/deathPlace>
    <http://dbpedia.org/resource/Palo_Alto,_California>

<http://dbpedia.org/resource/Steve_Jobs>
```

```
<http://schema.org/deathDate>
"5 de Outubro de 2011"
```

Conseguimos extrair as triplas que representam as relações das informações contidas naquele pequeno texto do Steve Jobs. Isso prova que toda a relação desses dados é compreendida por seres humanos. Ainda aproveitando essa sintaxe, as máquinas conseguem ler esses dados também.

8.4 Para ler mais

- Representação do Schema.org e seus vocabulários apresentados em formato RDFa — http://schema.org/docs/schema_org_rdfa.html
- RDFa — https://rdfa.info/play/

- ADIDA, Ben; BIRBECK, Mark; MCCARRON, Shane; HERMAN, Ivan. *RDFa Core 1.1 - Third Edition.* 2015. Disponível em: https://www.w3.org/TR/rdfa-core.

- BECKETT, David; BERNERS-LEE, Tim; PRUD'HOMMEAUX, Eric; CAROTHERS, Gavin. *RDF 1.1 Turtle.* 2014. Disponível em: https://www.w3.org/TR/turtle/.

- BENSON, Edward. *A Quick Tutorial on the Turtle RDF Serialization.* 2008. Disponível em: http://haystack.csail.mit.edu/blog/2008/11/06/a-quick-tutorial-on-the-tutrle-rdf-serialization/.

- HERMAN, Ivan. *RDFa Tutorial*. 2010. Disponível em: http://pt.slideshare.net/ivan_herman/rdfa-tutorial.

- HERMAN, Ivan. *RDFA 1.1 with a rich snippet example.* 2011. Disponível em: https://www.w3.org/blog/2011/05/rdfa-11-with-a-rich-snippet-ex/.

Capítulo 9
JSON-LD

> *The current process for developing next generation standards for the Web is too unpredictable and too constrained by limited W3C resources.* **— Manu Sporny**

JSON-LD é o acrônimo de **JavaScript Object Notation for Linked Data**. Como muitos outros tipos de dados estruturados, o JSON-LD é um formato para representar os conceitos de Linked Data no formato JSON.

JSON é um formato muito simples de texto para estruturação de dados, fácil de ler e entender por seres humanos e, para as máquinas, muito simples de interpretar e gerar. Como ser humano, você consegue escrever facilmente a estrutura de dados manualmente usando JSON, por causa do seu formato composto por `chave: valor`.

9.1 O que é JSON

O JSON é um formato de texto que facilita a estrutura de troca de dados e pode ser usado com praticamente qualquer linguagem de programação. Seu formato foi baseado nos objetos literais do JavaScript e tem uma sintaxe muito enxuta, formada por duas partes principais: chave e valor.

- A **chave** sempre é uma string, isso quer dizer que ela sempre precisa estar entre aspas duplas.
- O **valor** pode ser uma string, um número, uma expressão booleana, uma array ou um objeto.

A chave sempre é dividida por : (dois pontos).

```
{
  "chave": "string"
}
```

Pela simplicidade da sintaxe e do uso entre sistemas, o formato JSON se tornou padrão de mercado para troca e uso de dados entre sistemas. O XML é conhecido por ser muito complexo de parsear e montar, por isso o JSON ganhou tanta popularidade. Veja um exemplo simples:

```
{
  "carros": {
    "honda": {
      "carro": {
        "nome": "Civic",
        "portas": 5,
        "tipo": "Sedan"
      },
      "carro": {
        "nome": "City",
        "portas": 5,
        "tipo": "Sedan"
      },
    },

    "chevrolet": {
      "carro": {
        "nome": "Cruze",
        "portas": 5,
        "tipo": "Sedan"
      },
      "carro": {
        "nome": "Prisma",
        "portas": 5,
        "tipo": "Sedan"
      },
    },

    "fiat": {
```

```
      "carro": {
        "nome": "Linea",
        "portas": 5,
        "tipo": "Sedan"
      },
      "carro": {
        "nome": "Palio",
        "portas": 3,
        "tipo": "Hatch"
      },
    },
  }
}
```

Um XML equivalente ficaria mais ou menos assim:

```
<?xml version="1.0" encoding="UTF-8" ?>
<carros>
  <honda>
    <carro>
      <nome>City</nome>
      <portas>5</portas>
      <tipo>Sedan</tipo>
    </carro>
  </honda>
  <chevrolet>
    <carro>
      <nome>Prisma</nome>
      <portas>5</portas>
      <tipo>Sedan</tipo>
    </carro>
  </chevrolet>
  <fiat>
    <carro>
      <nome>Palio</nome>
      <portas>3</portas>
      <tipo>Hatch</tipo>
    </carro>
  </fiat>
</carros>
```

Tipos de valores

Para estruturar dados, o valor do atributo em um JSON pode ter vários tipos de valores:

- **Booleano ou nulos**: true, false, null;
- **Número**: inteiros ou não inteiros;
- **String**: textos;
- **Array**: uma coleção de valores;
- **Objeto**: um array que forma uma coleção de chaves e valores.

Booleanos ou nulos

Você pode ter os valores `true`, `false` ou `null` para representar valores booleanos ou nulos. Nesses casos, os valores não deverão estar dentro de aspas duplas.

```
"carro": {
  "nome": true,
  "portas": false,
  "tipo": null
}
```

Arrays

A estrutura de arrays é formada com colchetes, com valores separados por vírgulas. Um array pode estar vazio. Veja um exemplo:

```
"honda": ["civic", "fit", "city"],
"fiat": ["linea", "palio", "mobi"],
"gurgel": []
```

Objetos

Objetos são formados pelos sinais de `{}` (chaves), contendo uma coleção de chaves e valores.

```
"carro": {
  "nome": "Civic",
  "portas": 5,
  "tipo": "Sedan"
}
```

As chaves `nome`, `portas` e `tipo` estão dentro de `carro`. Este é um exemplo de hierarquia em dados estruturados com JSON.

Números

A representação de números é similar a já utilizada nas linguagens de programação conhecidas. Um número é um inteiro que pode ser negativo, colocando o sinal de `-` (menos) no início, e pode ser seguido por uma fração. De acordo com a documentação oficial da ECMA, valores numéricos que não podem ser representados como uma sequência de dígitos não são permitidos (como *Infinity* ou *NaN*).

```
"civic": {
  "portas": 5,
  "comprimento": 4.525
}
```

String

Strings são textos Unicode. São envolvidos entre aspas duplas usando barras invertidas como caractere de escape, quando necessário.

```
"carro": {
  "nome": "City",
  "tipo": "Sedan"
}
```

Um exemplo simples

Da própria documentação do jQuery, temos o seguinte exemplo:

```
<!DOCTYPE html>
<html>
<head>
  <title>Exemplo para ler JSON</title>
  <meta charset="utf-8">
</head>
<body>


<script type="text/javascript" src="https://code.jquery.com/jquery-
3.1.1.min.js"></script>
<script type="text/javascript">

  $(function(){

    $.getJSON( "https://api.github.com/users/diegoeis", function( data ) {
      var items = [];
      $.each( data, function( key, val ) {
        items.push( "<li id='" + key + "'>" + key + ": " + val + "</li>"
);
      });

      $( "<ul/>", {
        "class": "my-new-list",
        html: items.join( "" )
      }).appendTo( "body" );
    });


  })

</script>

</body>
</html>
```

A ideia do exemplo é simples: ele pega um perfil no GitHub e extrai algumas informações desse perfil. Ali nos parênteses da chamada `$.getJSON` , coloquei o endereço da API do GitHub que recupera as

informações do perfil. Ele pega os valores de chave/valor do JSON, e joga em um objeto `items`.

Depois ele pega isso tudo e joga em uma lista que será mostrada no `body`. Se você jogar esse código em um arquivo HTML, você verá o código funcionando.

Agora que já conhecemos um pouco sobre JSON, vamos falar sobre JSON-LD.

9.2 Começando JSON-LD

Agora que temos um *overview* sobre JSON e sua sintaxe, podemos começar a falar sobre JSON-LD. Sempre que queremos estender as informações do nosso código HTML, usamos algum tipo de dados estruturados. O mais comum é adicionar algum tipo de atributo aos elementos do HTML, indicando que aquele trecho de informação tem um significado específico.

Veja um exemplo usando RDFa:

```
<div vocab="http://schema.org/" typeof="Person">
  <a property="image"
href="http://2.gravatar.com/avatar/1bf877955dc2e43662320fd3b0280166">
    <span property="name">Diego Eis</span></a>,
  <span property="jobTitle">Coordenador de Times de Produtos</span>
  <div>
    E-mail: <a property="email"
href="mailto:diego@tableless.com.br">diego@tableless.com.br</a>
  </div>
  <div>
    Link: <a property="url" href="http://diegoeis.com">Blog do Diego</a>
  </div>
</div>
```

O problema é que isso depende da modificação do código HTML, o que muitas vezes pode ser difícil de fazer no seu projeto. O HTML

pode ser gerado automaticamente por algum CMS, framework etc. Imagine quando seu sistema ou website passar por um redesign, onde todo o HTML é modificado. Se os responsáveis pelo front-end não forem atentos ou não combinarem previamente, com certeza a marcação do HTML não será completa.

Por conta desses eventuais problemas, o JSON-LD se torna uma opção muito interessante, porque ele não depende da modificação do HTML, mas trata apenas de servir uma versão JSON dos dados importantes da página. Isso retira toda dependência da informação exibida pelo HTML ou de outros fatores que possam dificultar o acesso à informação.

Para implementar, basta simplesmente colocar em cada página importante do seu site/sistema, um pedaço de JSON que servirá os dados relevantes daquela página, juntamente com outras informações importantes que possam ser úteis. Veja um exemplo do meu próprio blog:

```
<script type="application/ld+json">
  {
    "@context":"http://schema.org",
    "@type":"BlogPosting",
    "@id":"http://diegoeis.com/um-pouco-sobre-a-experiencia-de-ir-
trabalhar-sem-carro.html",
    "author" : {
      "@type" : "Person",
      "@id":"http://diegoeis.com",
      "name" : "Diego Eis"
    },
    "publisher": {
      "@type":"Person",
      "name":"Diego Eis"
    },

"image":"https://pbs.twimg.com/profile_images/570009494805946368/dfEcOkau.
jpeg",
    "datePublished": "2016-09-22 00:00:00 UTC",
    "dateModified": "2016-09-22 00:00:00 UTC",
```

```
    "headline":"Um pouco sobre a experiência de ir trabalhar sem carro",
    "articleBody":"Aqui vai o texto do artigo."
  }

</script>
```

Esse código é geralmente usado em páginas de artigos, nas quais damos os dados importantes sobre artigo, bem como seu conteúdo, autor, datas etc.

Vamos entender um pouco mais a estrutura do JSON-LD mais à frente. Por enquanto, saiba apenas que coloco esse pedaço de script no `head` do meu HTML. Repare no detalhe do atributo `type` na tag `script`, que indica que esse pedaço de código é do tipo `json-ld`.

9.3 @context e @type

Lembra-se sobre o contexto de diálogos? Quando duas máquinas tentam trocar informações, elas precisam de um contexto para entender o vocabulário envolvido no diálogo com as outras máquinas.

Quando servimos um JSON-LD, damos esse contexto pelo atributo reservado `@context`. As chaves com nomes reservados são indicadas pelo sinal `@` (arroba), como `@id`, `@type` e outros. No atributo `@context`, indicamos o vocabulário para as máquinas, avisando qual o contexto em que elas interpretarão as chaves e os valores da página.

```
{
    "@context": "http://schema.org",
}
```

Nesse exemplo, usaremos o vocabulário do **Schema.org** por ser mais comum e fácil de assimilar. Como o Schema.org é uma biblioteca que contém vários tipos de vocabulários, com uma série

de opções, também indicaremos qual é o tipo de vocabulário que estamos usando. Para isso, basta indicar com o atributo reservado `@type`:

```
{
    "@context": "http://schema.org",
    "@type": "Person"
}
```

No caso desse nosso exemplo, qualquer máquina, robô, dispositivo ou outro meio que precisar recuperar informações sobre uma determinada página vai saber que deverá procurar no Schema.org o vocabulário de tipo `person`. Veja em http://schema.org/Person/ todas as propriedades que poderemos usar no atributo `type`.

9.4 Ambiguidade de dados e o @id

Dissemos com o código anterior que estamos nos referindo a uma pessoa, mas quem é essa pessoa? Para nomeá-la, vamos usar o atributo `name`:

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "name": "Diego Eis"
}
```

Mas aqui já começamos a esbarrar naquele problema de "ambiguidade" de dados. No meu site, posso estar usando **Diego Eis** para me identificar. Mas outra fonte de informação pode estar usando meu *username* `diegoeis`, assim:

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "name": "diegoeis"
}
```

Estou me referindo à mesma pessoa de duas formas diferentes. Logo, não podemos diferenciar uma fonte de informação pelo nome. É aí que entra uma das regras do conceito de Linked Data: usar URIs para nomear as coisas.

As URIs servem para identificar coisas. É como se fosse um RG, em que cada fonte tem sua identificação única. A URI que usamos aqui é a URL, que serve para duas coisas: identificar e localizar.

> **LEMBRE-SE**
>
> Como dito anteriormente, URI não é a mesma coisa de URL. URL é uma URI, e nem toda URI é uma URL.

Logo, vamos identificar uma fonte de informação utilizando sua URL na internet, que também serve para localizar essa fonte. Assim resolvemos o problema de ambiguidade:

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "@id": "http://diegoeis.com",
    "name": "Diego Eis"
}
```

Agora, tanto faz o que estiver escrito no atributo `name`, as máquinas saberão que eles estão falando da mesma pessoa, identificada pelo atributo `@id`.

O código anterior diz que estamos falando sobre uma **pessoa**, com nome de **Diego Eis**, identificado pela URI **http://diegoeis.com**.

Relacionando com outras fontes

Existem algumas chaves que nos permitem relacionar uma fonte com outra. Nós podemos dizer que um site, uma página ou uma pessoa tem algum tipo de relacionamento com outro site, página ou

pessoa. Nós também podemos dizer de que tipo é esse relacionamento. O nome que damos para isso é **embedding**.

É possível fazer algo assim:

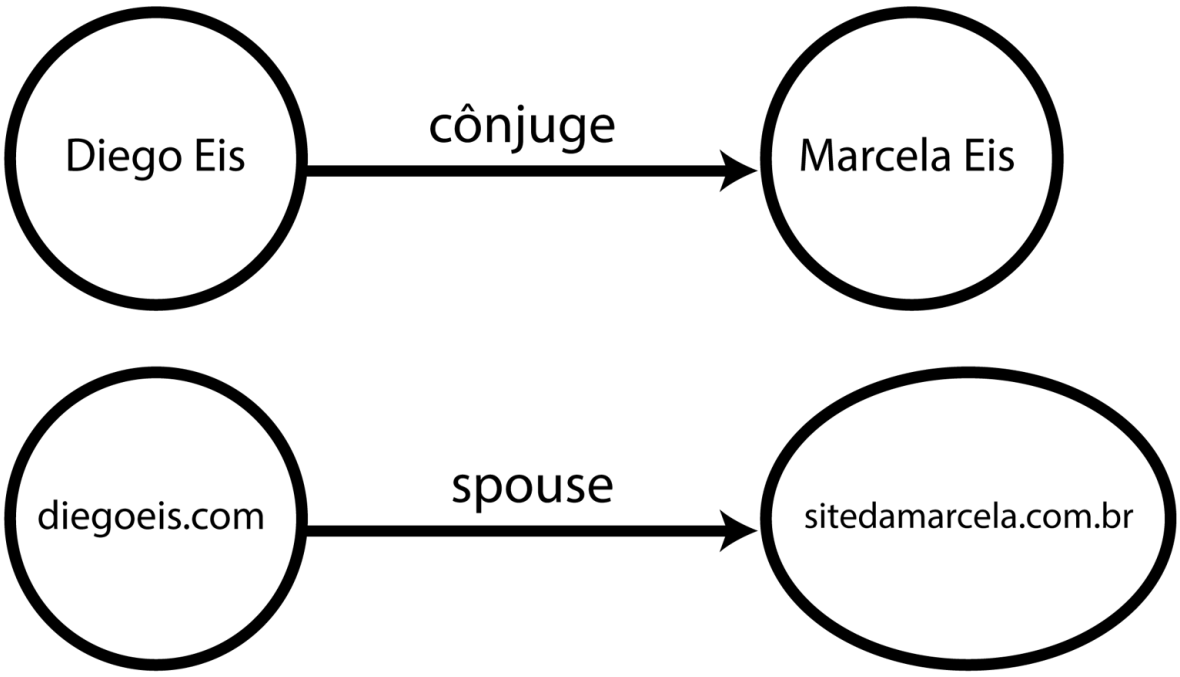
```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "@id": "http://diegoeis.com",
    "name": "Diego Eis",
    "knows": {
        "name": "Marcela Eis",
        "@id": "http://sitedamarcela.com.br"
    }
}
```

Isso forma um vínculo entre uma fonte e outra. Para melhorar ainda mais o exemplo, podemos dizer que a Marcela é minha esposa, usando a propriedade `Spouse` do vocabulário de `Person` do Schema.org.

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "@id": "http://diegoeis.com",
    "name": "Diego Eis",
    "spouse": {
        "name": "Marcela Eis",
        "@id": "http://sitedamarcela.com.br"
    }
}
```

Veja que eu não preciso indicar que minha esposa tem o `@type` `Person` , já que a propriedade `spouse` faz parte do vocabulário `Person` . Isso fica subentendido por causa do contexto. Já que estamos falando sobre a pessoa do “Diego Eis”, fica lógico que ele tem um cônjuge que é uma `Pessoa` . Ele não poderia ter um objeto (uma coisa) como cônjuge.

Podemos transformar esses dados que usamos como exemplo, em um grafo, assim como fizemos com RDF. Perceba que qualquer coisa que usa uma estrutura baseada em **triplas** pode ser transformada em um grafo. Nesse grafo, mostra que **Diego Eis** tem uma relação de **cônjuge** com **Marcela Eis**. Ficaria mais ou menos assim:



Espera-se que o site relacionado (no caso, o site da Marcela) sirva também dados em JSON-LD, para que as máquinas encontrem mais informações e assim o ciclo fique completo. No nosso exemplo, coloquei como `@id` o que seria o site da pessoa, mas poderia ser o link do perfil no Facebook, Twitter ou qualquer coisa que possa identificá-la. Para isso, podemos usar a propriedade `@sameAs`.

9.5 Propriedade sameAs

Uma pessoa, empresa ou website pode ter outras fontes de dados como suas redes sociais, por exemplo. Um site, como o Tableless,

pode ter vários outros canais que indicam sua propriedade e que podem ser identificados como a sua marca. Isso também pode causar um cenário no qual uma fonte de informação é relacionada por `@id` s diferentes.

Logo, é interessante que possamos dizer que o meu perfil do Twitter e do Facebook são tão válidos quanto meu website, ou que vários outros sites identifiquem uma empresa.

```
{
    "@context": "http://schema.org",
    "@type": "Person",
    "@id": "http://diegoeis.com",
    "name": "Diego Eis",
    "sameAs": [
        "http://facebook.com/diegoeis",
        "http://twitter.com/diegoeis",
        "http://instagram.com/diegoeis/"
    ],
    "spouse": {
        "name": "Marcela Eis",
        "@id": "http://sitedamarcela.com.br"
    }
}
```

Imagine que você tenha o seguinte texto:

```
Meu nome é Diego Eis. Esse é o meu perfil no facebook. Tenho também um
perfil no twitter e no instagram. Eu tenho uma esposa chamada Marcela Eis.
```

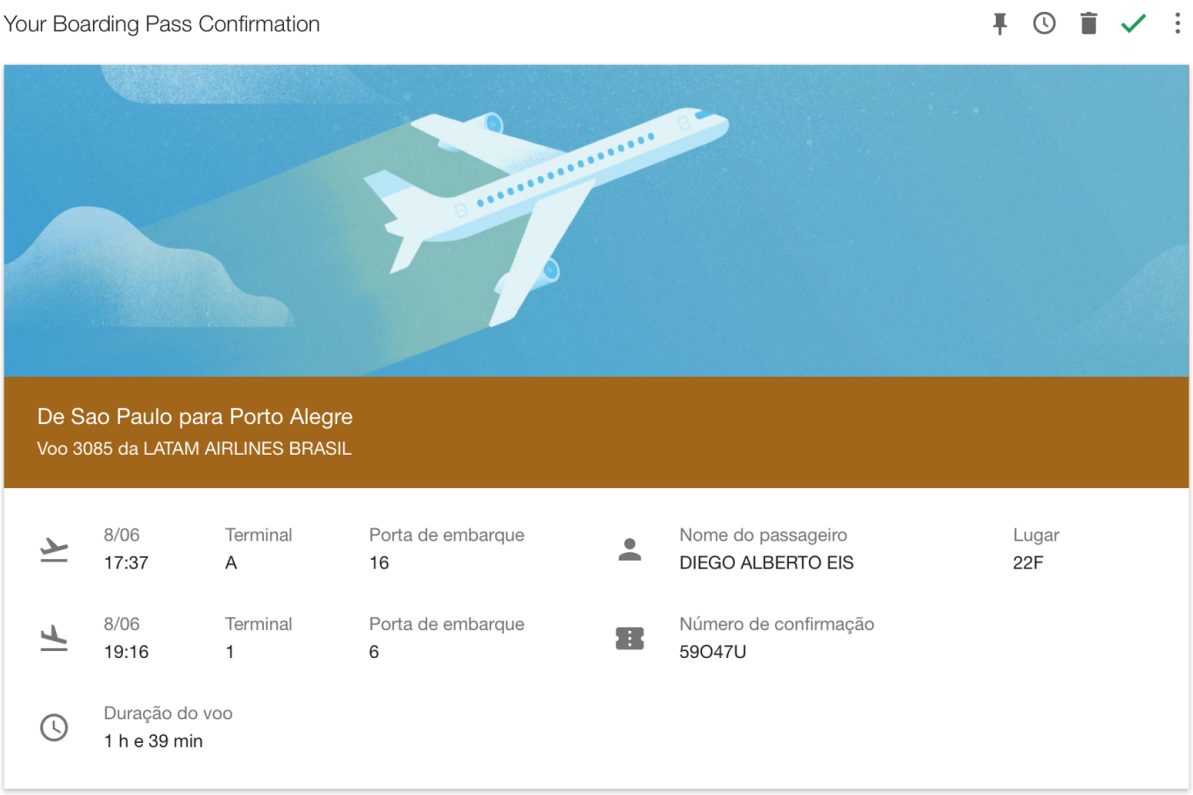
Marcando agora esse texto com RDFa, o exemplo poderia ser exatamente assim:

```
<div xmlns="http://www.w3.org/1999/xhtml"
  prefix="
    schema: http://schema.org/
    rdf: http://www.w3.org/1999/02/22-rdf-syntax-ns#
    rdfs: http://www.w3.org/2000/01/rdf-schema#">
  <div typeof="schema:Person" about="http://diegoeis.com/">
    <p>Meu nome é <span property="schema:name" content="Diego Eis">Diego
Eis</span>. Esse é o meu perfil no <span rel="schema:sameAs"
```

```
resource="http://facebook.com/diegoeis">facebook</span>. Tenho também um
perfil no
    <span rel="schema:sameAs"
resource="http://twitter.com/diegoeis">twitter</span> e no <span
rel="schema:sameAs"
resource="http://instagram.com/diegoeis/">instagram</span>.
    Eu tenho uma  esposa chamada <span rel="schema:spouse"
typeof="rdfs:Resource" about="http://sitedamarcela.com.br/"
property="schema:name" content="Marcela Eis">Marcela Eis</span>.
    </p>
  </div>
</div>
```

9.6 Quem está usando JSON-LD hoje?

Se você usa algum produto do Google, principalmente o Gmail ou Inbox, já deve ter percebido que, com o passar do tempo, sua caixa de entrada ficou mais inteligente. Dependendo das mensagens que você recebe, como por exemplo mensagens de agendamento de viagens, o e-mail da companhia aérea deve ter vindo como algo parecido com isso:



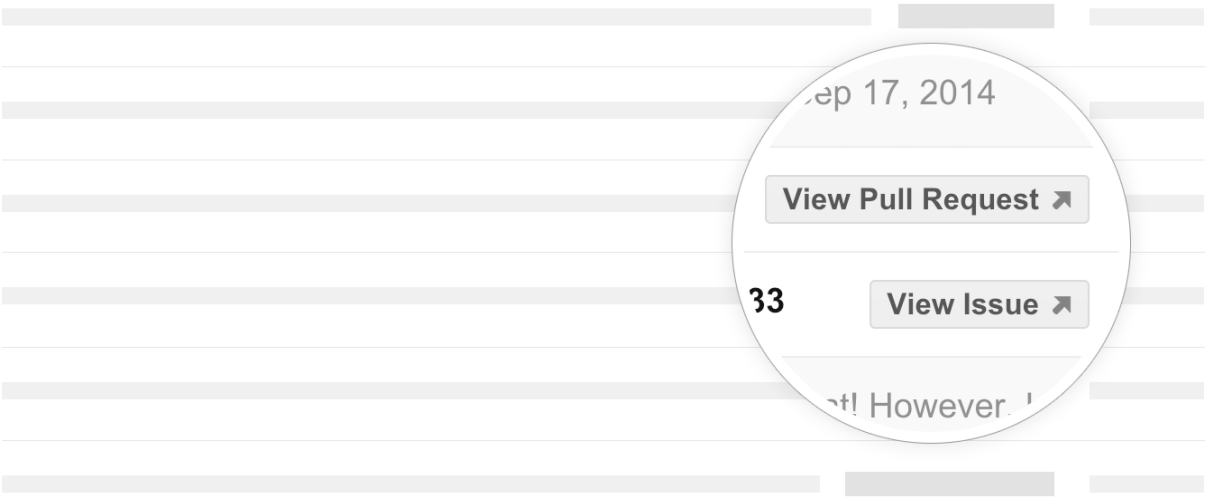
Todas essas informações são enviadas usando JSON-LD. O Inbox ou Gmail parseiam esses dados e formatam bonitinho, para que você tenha uma experiência mais rica nos seus e-mails.

O formato do JSON enviado para o Google seria mais ou menos esse:

```
{
  "@context": "http://schema.org",
  "@type": "Event",
  "name": "Taco Night",
  "startDate": "2015-04-18T15:30:00Z",
  "endDate": "2015-04-18T16:30:00Z",
  "location": {
    "@type": "Place",
    "address": {
      "@type": "PostalAddress",
      "name": "Google",
      "streetAddress": "24 Willie Mays Plaza",
      "addressLocality": "San Francisco",
      "addressRegion": "CA",
```

```
      "postalCode": "94107",
      "addressCountry": "USA"
    }
  },
  "potentialAction": [
    {
      "@type": "RsvpAction",
      "handler": {
        "@type": "HttpActionHandler",
        "url": "http://mysite.com/rsvp?eventId=123&value=yes"
      },
      "attendance": "http://schema.org/RsvpAttendance/Yes"
    },
    {
      "@type": "RsvpAction",
      "handler": {
        "@type": "HttpActionHandler",
        "url": "http://mysite.com/rsvp?eventId=123&value=no"
      },
      "attendance": "http://schema.org/RsvpAttendance/No"
    },
    {
      "@type": "RsvpAction",
      "handler": {
        "@type": "HttpActionHandler",
        "url": "http://mysite.com/rsvp?eventId=123&value=maybe"
      },
      "attendance": "http://schema.org/RsvpAttendance/Maybe"
    }
  ]
}
```

Alguns e-mails podem ter ações. Por exemplo, quando você recebe um e-mail do GitHub, digamos, sobre um Pull Request, você visualiza um botão para acessar esse Pull Request.



O Google tem implementado esse tipo de *feature* em vários dos seus serviços. Você pode ver como fazer e-mails mais inteligentes em https://developers.google.com/gmail/markup/, que o Google preparou para os desenvolvedores. Ele não só aceita JSON-LD, mas também vários outros formatos, como Microdata. Contudo, o JSON-LD é o queridinho do Google e de outras empresas.

O Tableless usa para melhorar o SEO e contribuir com a distribuição de dados na internet. Entre em http://bit.ly/2cUD9Ur para ver na ferramenta do Google os dados que servimos no nosso site. Estamos usando o vocabulário do Schema.org do tipo `Article` (http://schema.org/Article).

Como não há necessidade de modificações profundas no seu HTML, você pode servir os dados em JSON-LD no seu site hoje mesmo. Se você tem um blog ou um site de conteúdo, o benefício direto será o SEO. Além de deixar as informações do seu site públicas para qualquer máquina acessar.

9.7 Para ler mais

- RDF Translator — http://rdf-translator.appspot.com

- Ferramenta do Google para testar Dados Estruturados — https://search.google.com/structured-data/testing-tool/
- Steal Our JSON-LD — http://jsonld.com
- Editor Online de JSON — http://www.jsoneditoronline.org
- Especificação da ECMAScript 2016 — http://www.ecma-international.org/ecma-262/7.0/index.html
- CROCKFORD, D. *The application/json Media Type for JavaScript Object Notation (JSON).* 2006. Disponível em: https://www.ietf.org/rfc/rfc4627.txt.
- KELLOGG, Gregg. *Building JSON-LD APIs:* Best Practices. 2016. Disponível em: http://json-ld.org/spec/latest/json-ld-api-best-practices/.
- MEDEIROS, Ícaro. *Linked data in use.* 2014. Disponível em: http://www.icaromedeiros.com.br/linked-data-in-use.html.

Capítulo 10
Extra: Turtle e SPARQL

A Web Semântica tem muito conteúdo técnico que eu não abordei no livro exatamente por ser um mundo muito extenso. Mas queria introduzir a você dois assuntos que podem ser o ponto de partida para uma segunda fase de estudos, para algo mais técnico: o Turtle e o SPARQL.

O Turtle é uma serialização do RDF para representação de triplas. O SPARQL é uma linguagem de query para extrair informações de fontes de dados baseadas em triplas.

10.1 O básico do básico sobre Turtle

Existe um monte de serializações do RDF, como N-Triples, Notation 3, N-Quads e outras. Eu não quero dar ênfase para todas elas porque tenho quase certeza de que não haverá uso prático nos seus projetos diários. Mas gostaria de dar um pitaco sobre o Turtle, que talvez possa ajudá-lo, assim como me ajudou, a entender todo esse esquema de relação de triplas.

Como vimos, o RDF é apenas um framework, ele é um modelo de abstração. O RDF é escrito baseando-se na sintaxe do XML, que é a representação oficial do W3C para explicar o RDF. O Turtle é uma maneira de serializar o RDF, com uma linguagem muito mais amigável, legível e fácil de editar do que o XML.

O RDF por si só não tem uma maneira de expressar triplas. Um documento Turtle permite representar um grafo RDF em uma forma compacta, baseada em texto.

O Turtle permite que substituamos as grandes URLs que usamos para identificar nossos dados, por um namespace amigável. Ainda com o exemplo do Steve Jobs, nós definimos um prefixo logo no início do código (que é nosso contexto, onde vamos definir o vocabulário) e depois modificamos o endereço dos predicados por esse prefixo. Teríamos o seguinte código:

```
@prefix schema:<http://schema.org/> .

<http://dbpedia.org/resource/Steve_Jobs>
    schema:alternateName
    "Steve Jobs"

<http://dbpedia.org/resource/Steve_Jobs>
    schema:spouse>
    <http://dbpedia.org/resource/Laurene_Powell_Jobs>

<http://dbpedia.org/resource/Steve_Jobs>
    schema:owns
    <http://dbpedia.org/resource/Apple_Inc.>

<http://dbpedia.org/resource/Steve_Jobs>
    schema:birthPlace
    <http://dbpedia.org/resource/San_Francisco>

<http://dbpedia.org/resource/Steve_Jobs>
    schema:birthDate
    "24 de Fevereiro de 1955"

<http://dbpedia.org/resource/Steve_Jobs>
    schema:deathPlace
    <http://dbpedia.org/resource/Palo_Alto,_California>

<http://dbpedia.org/resource/Steve_Jobs>
    schema:deathDate
    "5 de Outubro de 2011"
```

Esse código seria colocado em um arquivo `.ttl` e servido separadamente. O W3C mostra alguns exemplos de diferentes maneiras de escrever URLs no Turtle:

```
# Uma tripla com todas as URLs absolutas, como no nosso primeiro exemplo
<http://one.example/subject1> <http://one.example/predicate1>
<http://one.example/object1> .

@base <http://one.example/> .
<subject2> <predicate2> <object2> .     # URLs relativa, ex.
http://one.example/subject2

BASE <http://one.example/>
<subject2> <predicate2> <object2> .     # URLs relativa, ex.
http://one.example/subject2

@prefix p: <http://two.example/> .
p:subject3 p:predicate3 p:object3 .     # prefixo de nome, ex.
http://two.example/subject3

PREFIX p: <http://two.example/>
p:subject3 p:predicate3 p:object3 .     # prefixo de nome, ex.
http://two.example/subject3

@prefix p: <path/> .                    # prefixo p: agora se refere a
http://one.example/path/
p:subject4 p:predicate4 p:object4 .     # prefixo de nome, ex.
http://one.example/path/subject4

@prefix : <http://another.example/> .    # prefixo vazio
:subject5 :predicate5 :object5 .        # prefixo de nome, ex.
http://another.example/subject5

:subject6 a :subject7 .                 # atributo 'same as' :subject6
<http://www.w3.org/1999/02/22-rdf-syntax-ns#type> :subject7 .
```

Com uma linguagem de query, as máquinas podem vasculhar bancos de dados que estão estruturados por meio de triplas para conseguir extrair informações úteis. Uma dessas linguagens é o SPARQL.

10.2 O básico do básico sobre SPARQL

O SPARQL é uma linguagem de query para realizar consultas em dados estruturados usando o padrão RDF. O termo SPARQL é um acrônimo recursivo que significa *SPARQL Protocol and RDF Query Language*.

Uma query SPARQL consiste em uma estrutura simples de duas cláusulas: `SELECT` e `WHERE`. O **`SELECT`** identifica as variáveis que aparecerão nos resultados da query e o **`WHERE`** mostra o padrão básico do grafo que bate com os dados.

No exemplo a seguir, pego todas as obras de Mozart. Estou usando para esse exemplo um banco de dados de triplas RDF online do DBPedia. Se você quiser testar, entre na interface do iSPARQL da própria DBPedia. É uma interface bem feinha, não se espante: http://dbpedia.org/isparql/.

Segue o exemplo:

```
SELECT ?musics
WHERE {
  ?musics dbo:musicComposer dbr:Wolfgang_Amadeus_Mozart
}
```

A resposta que recebi em XML no formato RDF foi:

```
<rdf:RDF xmlns:res="http://www.w3.org/2005/sparql-results#"
xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
<rdf:Description rdf:nodeID="rset">
<rdf:type rdf:resource="http://www.w3.org/2005/sparql-results#ResultSet"/>
    <res:resultVariable>musics</res:resultVariable>
    <res:solution rdf:nodeID="r0">
      <res:binding rdf:nodeID="r0c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Teorema_(film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r1">
      <res:binding rdf:nodeID="r1c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Balthus_Through_the_Looking_Glas
```

```
s"/></res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r2">
      <res:binding rdf:nodeID="r2c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Teaching_to_See"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r3">
      <res:binding rdf:nodeID="r3c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Écoute_voir"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r4">
      <res:binding rdf:nodeID="r4c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/A_Man_Escaped"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r5">
      <res:binding rdf:nodeID="r5c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Ehrengard"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r6">
      <res:binding rdf:nodeID="r6c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Tale_of_Tales_(1979_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r7">
      <res:binding rdf:nodeID="r7c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/The_Magic_Flute_(1975_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r8">
      <res:binding rdf:nodeID="r8c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/The_Magic_Flute_(2006_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r9">
```

```
      <res:binding rdf:nodeID="r9c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Quiet_Night_In"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r10">
      <res:binding rdf:nodeID="r10c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Live_Together,_Die_Alone"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r11">
      <res:binding rdf:nodeID="r11c0"><res:variable>musics</res:variable>
<res:value rdf:resource="http://dbpedia.org/resource/Eroica_(2003_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r12">
      <res:binding rdf:nodeID="r12c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Killing_Time_(2013_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r13">
      <res:binding rdf:nodeID="r13c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Don_Giovanni_(1979_film)"/>
</res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r14">
      <res:binding rdf:nodeID="r14c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/Under_the_Pavement_Lies_the_Stra
nd"/></res:binding>
    </res:solution>
    <res:solution rdf:nodeID="r15">
      <res:binding rdf:nodeID="r15c0"><res:variable>musics</res:variable>
<res:value
rdf:resource="http://dbpedia.org/resource/A_Woman_Is_a_Risky_Bet:_Six_Orch
estra_Conductors"/></res:binding>
    </res:solution>
  </rdf:Description>
</rdf:RDF>
```

Lá você consegue ver a descrição das triplas, a query e também a resposta XML.

Tente pegar, por exemplo, todos os artistas relacionados ao John Lennon. Coisa simples, assim:

```
SELECT ?artists
WHERE {
  ?artists dbo:associatedMusicalArtist dbr:John_Lennon
}
```

Ou as músicas escritas por ele:

```
SELECT ?musics
WHERE {
  ?musics dbp:artist dbr:John_Lennon
}
```

Há uma série de combinações de filtros e comandos que você pode usar com SPARQL para combinar e extrair informações. O pessoal da Wiki Base22 tem uma documentação (https://wiki.base22.com/display/btg/SPARQL+Query+Examples) com exemplos muito boa. Lá você vai aprender a fazer perguntas usando esses dados.

As declarações RDF (as triplas) são apenas um framework, um padrão, para serem usadas em vários outros formatos como JSON-LD, Turtle, N-triples etc. A ideia é que não importa qual o formato você esteja usando para expressar os dados, a informação ainda será representada e organizada usando o trio: sujeito, predicado e objeto.

O W3C dá um exemplo bem legal para entender isso: números são o oposto de numerais. Números são um conceito matemático. Já os numerais são uma representação usando romanos, arábicos, hexadecimal etc. Se você vai mostrar os números usando o formato romano, não importa.

O RDF também não é uma aplicação XML. O modelo fundamental do RDF é totalmente independente do XML. Ele é um modelo que descreve e qualifica relações entre duas fontes web.

A página do *John Lennon* lá na DBPedia (http://dbpedia.org/page/John_Lennon) tem todas as informações sobre o cantor. Você consegue essas informações em diversos formatos, incluindo XML em padrão RDF (http://dbpedia.org/data/John_Lennon.rdf).

Esse arquivo RDF tem algo em torno de 2.148 linhas, com todos os dados que a DBPedia tem de John Lennon. Há outros formatos disponíveis, como JSON-LD (http://dbpedia.org/sparql?default-graph-uri=http%3A%2F%2Fdbpedia.org&query=DESCRIBE%20%3Chttp%3A%2F%2Fdbpedia.org%2Fresource%2FJohn_Lennon%3E&format=application%2Fjson-ld).

Capítulo 11
Polêmica: código limpo ainda é necessário?

Fique calmo e tente não me xingar ao terminar de ler a próxima afirmação. Quero deixar claro que essa é minha opinião e, sim, 90% dos desenvolvedores vão discordar de mim, mas **código front-end limpo não será mais importante**.

Escrever código limpo é uma regra básica para qualquer programador desde o primeiro dia em que a primeira linha de código foi escrita. Existem vários motivos para ter um código bem estruturado e legível, como:

1. Código ruim gera mais manutenção;
2. Código bom é documentação;
3. Código ruim, geralmente, é difícil de testar;
4. Código ruim, geralmente, é redundante;
5. Código ruim é difícil de ser estendido;
6. Código ruim é difícil de ser entendido por outros programadores;
7. Código ruim gera dependências ruins;

Quando falamos sobre código front-end, todos esses motivos também se aplicam para motivar a escrita de um código limpo e legível. Mas código limpo só é importante se esse código é gerado por um ser humano.

Desde sempre os front-ends reclamavam sobre o código gerado automaticamente por programas como o Dreamweaver. Eles tinham uma razão para isso: o código gerado era péssimo. Era um tempo em que a conexão com a internet era precária e tudo o que pudéssemos fazer para melhorar o carregamento do site, nós fazíamos. O código gerado por programas Wysiwyg tinha vários problemas: era difícil de ler, não havia semântica alguma, continha código inútil e, muitas vezes, não era compatível com todos os

browsers. Tudo isso fazia com que o código limpo, semântico, enxuto e acessível tivesse um valor inestimável.

Código limpo era sinônimo de bom ranking no Google, boa compatibilidade entre os browsers, performance de carregamento garantida, produtividade entre os membros do time por causa da legibilidade do código, facilidade de manutenção etc. Então, escrever código limpo era uma premissa para que seu site funcionasse bem. Escrever código limpo era uma necessidade não apenas para que seu amiguinho do lado soubesse o que estava sendo programado, mas principalmente para fazer seu sistema funcionar direito na interweb.

Hoje, boa parte dos problemas citados foi resolvida e não por causa do código bonito, mas por causa do avanço das tecnologias envolvidas:

- Os browsers têm uma ótima complacência com os padrões web, extinguindo a maioria dos problemas de compatibilidade de layout;
- A performance é atacada em várias frentes: processo de build dos assets, tecnologias como HTTP/2, e até a evolução da conexão que fica mais rápida a cada ano;
- A manutenção e a legibilidade do código HTML/CSS não são mais um problema sério, já que o HTML é facilmente escrito usando poucas tags e o CSS tem os pré-processadores que auxiliam muito na hora de definir padrões, além das boas práticas;
- O JS está bem assessorado por frameworks, libraries e uma série de boas práticas que se responsabilizam pela parte pesada do trabalho, deixando pouca margem de erro para os devs;
- E o mais importante para mim é que a semântica não está mais no HTML. Desde a vinda de tecnologias com o JSON-LD, a semântica não está mais atrelada ao código HTML, e isso é ótimo.

Além disso, uma série de ferramentas foram criadas para nos auxiliar em tarefas do cotidiano. Nós já usamos coisas como Grunt e Gulp para "buildar" nossos assets, deixando-os prontos para serem publicados.

Veja só: nós fazemos manualmente um código lindo, maravilhoso, para que depois essas ferramentas enfeitem o código para que ele possa ser mais performático. Máquina não precisa de código bonito, quem precisa de código bonito é você e seu time para entenderem melhor o que foi escrito. Nem código semântico nós precisamos escrever mais, dado que, com tecnologias como JSON-LD, a semântica está cada vez mais agnóstica, sem a necessidade de estar presa ao código de marcação.

Na minha opinião, em um cenário futuro bem próximo, código gerado automaticamente por diversos serviços, programas e ferramentas (o Sketch, para começar) será maioria. Cada vez mais os desenvolvedores front-end escreverão código original.

11.1 WebAssembly (Wasm)

Outro ponto importante que está avançando cada vez mais rápido é toda aquela história do WebAssembly. O WebAssembly é um novo formato binário criado pelo Google, Microsoft, Mozilla e vários outros.

O formato de código binário do WebAssembly pode ser decodificado muito mais rápido do que o JavaScript é parseado. Isso realmente traz para a Web a experiência de programas nativos, principalmente no mobile.

O legal é que outras linguagens podem ser compiladas para WebAssembly. Hoje o projeto está um pouco mais focado em C/C++, mas com certeza outras linguagens serão alcançadas a partir daí. O objetivo principal do WebAssembly é a performance.

Uma preocupação que surge no ar é que isso cheira a monopólio. Lembra do Flash? Querendo ou não, ele era uma alternativa de criar algo nativo na Web. Mas a graça é que:

1. WebAssembly não substitui o JavaScript. Tudo tem retrocompatibilidade; tudo será executado no mesmo universo que o JS e a segurança terá as mesmas restrições que o JS.
2. Não é só uma empresa ou um grupo que está por trás do Wasm, mas várias como Firefox, Chromium, Edge e Webkit.
3. Para rodar WebAssembly, não será necessário rodar plugins de terceiros, já que os motores dos browsers serão totalmente compatíveis.

Imagine agora que ferramentas como Sketch, ou até serviços como Wix, gerem código automaticamente e publiquem nativamente com WebAssembly na Web.

11.2 O front-end como conhecemos pode morrer?

Sim. Eu escrevi um artigo bem polêmico (EIS, 2017a) sobre esse assunto e quase fui linchado pela comunidade. Talvez eu tenha escrito esse artigo muito cedo, e talvez eu deveria ter escrito de forma diferente. Mas a verdade é que, como qualquer outra profissão, o dev front-end pode sim morrer.

O front-end como conhecemos hoje, em que o dev faz todo o trabalho mecânico de passar o layout para HTML/CSS e depois usa uma biblioteca ou um framework para conversar com uma API qualquer, vai morrer com quase toda (90%, em minha humilde opinião) certeza. Contudo, ainda haverá espaço para um tipo de profissional que faz um trabalho mais lógico.

Logo adiante separei uma série de links que evidenciam o caminho que o mercado está tomando, e como algumas iniciativas e serviços estão automatizando código front-end.

11.3 Para ler mais

- BIGGINS, Matthew. *The Future of Web Development:* Coding as a Service. Mar. 2017. Disponível em: https://goo.gl/n2qbKL.
- EIS, Diego. *O fim da profissão front-end.* Mar. 2017a. Disponível em: http://diegoeis.com/o-fim-do-front-end.html.
- EIS, Diego. *The (probable) end of the front-end profession as you know it.* Mar. 2017b. Disponível em: https://medium.com/@diegoeis/the-probable-end-of-the-front-end-profession-as-you-know-it-2e205ecb4ab8.
- FINLEY, Klint. *The $11 M Tool That Could Help Computers Write Their Own Code*. Nov. 2014. Disponível em: https://www.wired.com/2014/11/darpa-pliny/.
- HUK, Andree. *With Sketch 43, Design IS Code & Code IS Design*. Abr. 2017. Disponível em: https://uxdesign.cc/design-is-code-code-is-design-b941c14f3fd8.
- MIHOV, Dimitar. *This app uses artificial intelligence to turn design mockups into source code.* Maio 2017. Disponível em: https://thenextweb.com/apps/2017/05/26/ai-raw-design-turn-source-code/#.tnw_E5IF7iEY.
- MILLER, Carson. *The future of front-end development is design*. Nov. 2016. Disponível em: https://techcrunch.com/2016/11/29/the-future-of-front-end-development-is-design/.

- MILLER, Meg. *This Startup Uses Machine Learning To Turn UI Designs Into Raw Code*. Jun. 2017. Disponível em: https://www.fastcodesign.com/90127911/this-startup-uses-machine-learning-to-turn-ui-designs-into-raw-code.
- TAMUZ, Yaron. *Sketch 43 will change the way we work and think*. Abr. 2017. Disponível em: https://medium.com/sketch-app-sources/sketch-43-will-change-the-way-we-work-and-think-e4c919957423
- TANZ, Jason. *Soon we won't program computers. We'll traind them like dogs:* The end of Code. Maio 2017. Disponível em: https://www.wired.com/2016/05/the-end-of-code/.
- WHEATLEY, Mike. *Startup uses AI to create programs from simple screenshots*. Maio 2017. Disponível em: https://siliconangle.com/blog/2017/05/28/startup-uses-ai-create-gui-source-code-simple-screenshots/.
- **Can computers write their own algorithms?** — https://www.quora.com/Can-computers-write-their-own-algorithms
- **Why can't there be a program that writes code for us?** — https://www.quora.com/Why-cant-there-be-a-program-that-writes-code-for-us
- **A.I. is Progressing Faster Than You Think!** — https://youtu.be/mQO2PcEW9BY
- **React Sketch.app - "render React components to Sketch"** — http://airbnb.design/painting-with-code/ e https://github.com/airbnb/react-sketchapp
- **You design, we put it on the web.** — https://launchpad.animaapp.com/DreamShareSample/home
- **Pix2Code** — https://youtu.be/pqKeXkhFA3I
- **Wix Code - Create advanced web applications without coding** — https://www.producthunt.com/posts/wix-code

Capítulo 12
Concluindo

> *Data is anything but "raw", that we shouldn't think of data as a natural resource but as a cultural one that needs to be generated, protected, and interpreted.* **— Lisa Gitelman (2013)**

Embora estejamos falando sobre Web durante todo o livro, a Semântica não se aplica apenas à Web. Na verdade, na minha opinião, as pessoas começarão a experimentar a utilidade e o que realmente é Semântica em outras plataformas e ambientes.

12.1 Excesso de informação e a decisão imediata

Todos nós gostamos de novidades. É comum que nos sintamos atraídos por qualquer coisa que seja novo e diferente do nosso cotidiano. Mesmo que você tenha uma rotina de ir e voltar ao trabalho, você está procurando algo novo nas redes sociais, na internet, em um livro, em uma música, ou até observando as pessoas à sua volta, prestando atenção nos seus comportamentos ou nas conversas alheias dentro do ônibus.

Sempre que tomamos essas doses diárias de novidade, nosso cérebro nos dá uma pequena recompensa em forma de dopamina, que nos traz satisfação sempre que conhecemos alguma novidade, por menor que esta seja. A dopamina é um neurotransmissor muito importante para o funcionamento do cérebro, mas que também está ligada a vários tipos de vícios. A dopamina encoraja que você faça alguma coisa repetidas vezes a fim de sentir novamente a recompensa de satisfação e prazer.

Existe um vídeo muito interessante com o Nicholas Carr (https://www.youtube.com/watch?v=PF1JgIWbSIQ) que explica o que exatamente a internet está fazendo com o nosso cérebro. Lá ele mostra como a procura pela novidade imediata está criando uma ânsia por novidades imediatas no meio digital, que antes não era comum.

O problema de querermos respostas e novidades imediatamente, o tempo inteiro, é que nós não estamos preparados para absorver tanta informação assim. Hoje nós não temos como filtrar e reter toda a carga de informação que recebemos dos diversos meios de comunicação, incluindo livros, televisão, rádio e principalmente internet.

Junto com essa quantidade de informação, vêm o imediatismo, onde precisamos ter respostas rápidas aos estímulos que damos no mundo digital (principalmente, nas redes sociais). Quando subimos uma foto, queremos logo saber quem viu, curtiu e comentou. Nós ficamos sempre aflitos pelo simples sentimento de que talvez estamos perdendo alguma coisa ou que estamos atrasados. Aí entra uma citação bem bacana do Wayne Luke:

> *O mal-estar de nosso tempo é a inadequa ão, o sentimento opressivo de que as outras pessoas estão fazendo as coisas certas, lendo os livros que contam e usando os computadores e programas mais modernos, enquanto nós estamos ficando para trás na carreira ou nos relacionamentos.*

Além da quantidade abundante das informações supérfluas que queremos consumir por prazer, ou informações que precisamos consumir para estudar ou adquirir algum conhecimento, há uma série de outras informações (geralmente totalmente descartáveis) que precisamos filtrar imediatamente para executar tarefas corriqueiras.

Atualmente, existem vários serviços que se propõem a melhorar o seu poder de escolha. O exemplo mais comum são serviços que o ajudam a planejar uma viagem, como o Google Trips, ou algo mais simples como o Trivago, que se propõe a agrupar, relacionar e categorizar uma grande quantidade de ofertas de quartos de hotéis, a fim de minimizar o seu trabalho na hora de escolher o hotel mais barato.

Serviços como estes não fazem nem a metade do trabalho que deveriam fazer na realidade. Digo isso porque eles apenas sintetizam todas as opções em um mesmo lugar, mas na realidade eles não o ajudam a escolher: quem disse que o hotel mais barato é o hotel que você vai gostar? Quem disse que um hotel 5 estrelas é o mais conveniente para a cidade que você vai viajar?

Mas como esses serviços fariam o ciclo completo? A tarefa de descobrir que você tem uma viagem agendada para uma cidade, comprar as passagens (com os assentos que você mais gosta), alugar o carro (na empresa que você prefere) e reservar o hotel (que você costuma ficar se for para um lugar de viagem frequente, ou um que se encaixe mais ou menos no seu gosto) não é trivial. Tudo depende de muitos fatores: o sistema não só precisa saber informações pessoais sobre você, mas ele também precisa saber conversar com os respectivos serviços para procurar esses hotéis, companhias aéreas e locadoras de carros. Para que isso aconteça, todas as pontas precisam falar no mesmo idioma, seus dados precisam estar em um mesmo formato e, principalmente, precisam estar acessíveis.

Se um sistema conseguir fazer tarefas desse tipo, relativamente complexas, mas corriqueiras, nós não precisaríamos lidar com informações excessivas, deixando para as máquinas tomarem algumas das decisões triviais.

> *O processo de transformar informa ão pura em algo com significado requer conectar os pontos entre informa ão limitada que faz sentido para você e o catálogo de modelos mentais, cren as, símbolos e associa ões que você guardou de experiências anteriores.* **— Buster Benson (2017)**

As máquinas têm mais capacidade de guardar e analisar informações históricas a fim de tomar decisões melhores no futuro. Quanto mais informação e decisões corretas são tomadas, mais capacidade as máquinas ganham de tomar boas decisões no futuro.

12.2 A semântica e a Internet das Coisas

Internet das Coisas (IoT) não é novidade, embora ainda não tenhamos dispositivos, casas e outros ambientes conectados e trocando informações entre si. E para ser sincero, acho que isso vai demorar alguns anos (não muitos) para acontecer. Não apenas porque as pessoas não estão preparadas — eu acho que elas estão —, mas por causa da própria tecnologia.

Eu ainda sofro para configurar direito um repetidor sem fio na minha própria casa. Eu tive de passar um fio de rede para que o sinal chegasse sem perdas no repetidor, para então fazê-lo funcionar como roteador padrão. A tecnologia de infraestrutura ainda não suporta todas as promessas que a IoT pode nos oferecer.

Para você que chegou agora nesse negócio de IoT, tentarei explicar um pouco: imagine um lugar no qual os dispositivos pudessem se conversar e trocar dados — e aqui eu quero que você expanda um pouco o conceito de "dispositivos". Não imagine um celular, mas imagine qualquer "coisa".

Os dados são coletados pelos próprios dispositivos ou por outros que trabalham como sensores ou fontes de dados do ambiente. O *stream* de dados também seria contínuo entre os dispositivos. Todos os dispositivos enviariam e receberiam dados de todos os outros dispositivos à sua volta, com o intuito de entender qual o status do seu ambiente.

Em um ambiente no qual todos os dispositivos à sua volta estarão permanentemente conectados, é necessário que a infraestrutura esteja preparada. Um monte de empresas está investindo em pesquisas para tentar resolver esse problema, tentando viabilizar tecnologias, como a rede 5G, para que a troca de petabytes de dados seja feita sem problemas. Imagine um cenário em que carros autônomos dependem da troca de dados entre outros carros, a rua e os semáforos. Sem uma rede decente, perder pacotes na transmissão de dados significa risco de acidentes fatais.

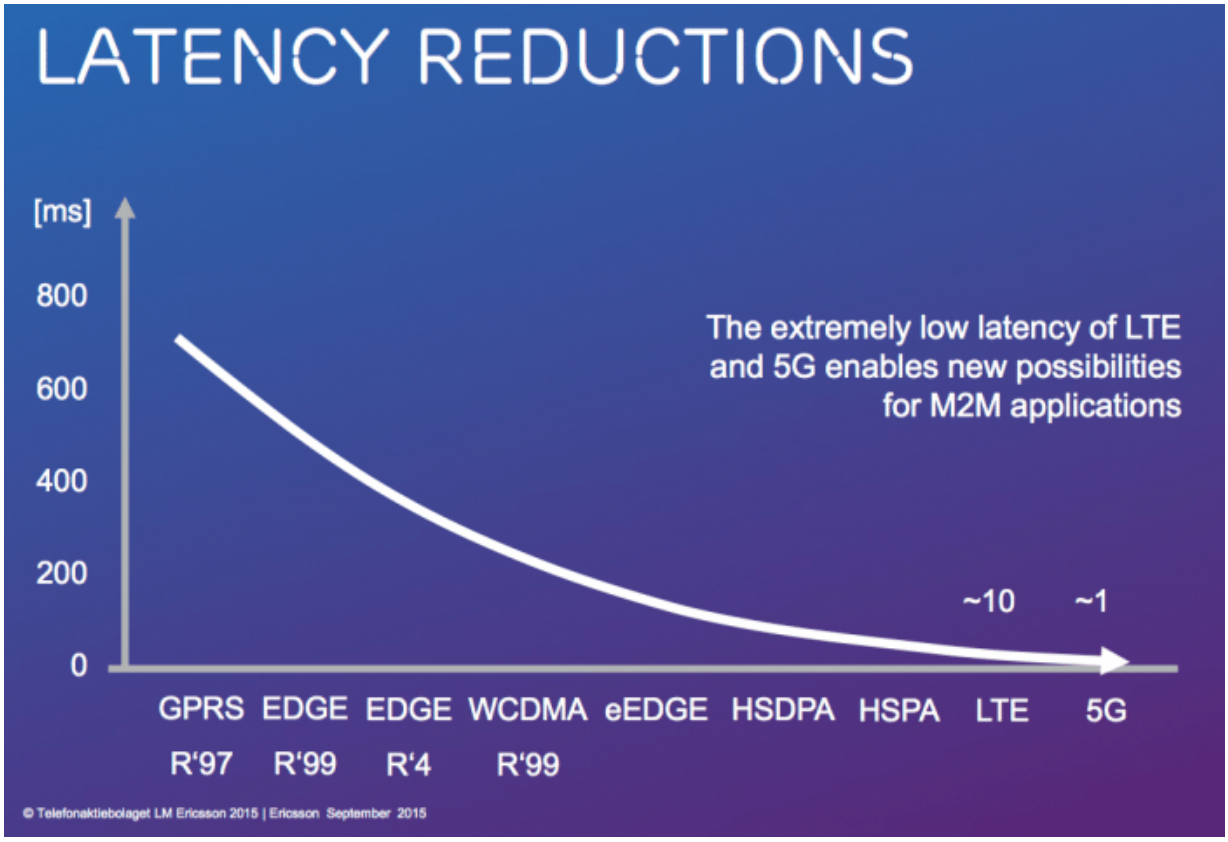


Figura 12.1: Fonte: https://tecnoblog.net/wp-content/uploads/2016/03/5g-ericsson-latencia-700x477.png

Além disso, há outras barreiras que precisam ser ultrapassadas, como segurança (ainda um tópico muito debatido e sensível quando se trata de Internet das Coisas), como também o ambiente no qual esses dados serão guardados, toda a infraestrutura envolvendo Clouds e outras tecnologias. Os conceitos da semântica estarão por todos os lados em um cenário ultraconectado como este, fazendo com que os dados possam ser de fato entendidos por todos os dispositivos:

- Os dados deverão ter notações semânticas;
- Os dados deverão ser de qualidade;
- Eles devem estar em formatos que sejam conhecidos;
- Os dados deverão ter a possibilidade de se interconectar e ter links para outros dados;

- Os dados devem ser adaptados de acordo com o contexto das soluções.

> *A ausência de interoperabilidade ao longo das plataformas de IoT acaba por provocar a fragmenta ão, além de silos de dados e custos maiores de desenvolvimento.* **— Dave Raggett, líder do grupo de Web of Things do W3C Internacional**

A era da Internet das Coisas, juntamente com a Web Semântica, vai se tratar de um ambiente no qual a informação dada é bem definida quanto ao seu significado, para possibilitar o entendimento desses dados por objetos, dispositivos e pessoas a fim que estes possam trabalhar em cooperação. Isso permitirá interações entre dispositivos e coisas de forma automática.

Isso quer dizer que tarefas, desde as mais simples até as mais complexas, poderão ser feitas por seres humanos, mas por máquinas, robôs etc. Uma coisa (celular, assistente pessoal tipo a Siri ou Cortana, Chatbot, um carro, um micro-ondas, qualquer coisa) vai navegar, procurar, filtrar e consolidar informações para uma pessoa ou para executar uma tarefa entre as outras máquinas.

Com a expansão da IoT, a interoperabilidade se torna cada vez mais importante. Organizações que desenvolvem padrões, como o W3C, já fizeram um grande trabalho ao criar tecnologias que usamos como padrão hoje em dia. Esse trabalho ainda continua com a necessidade da criação de novos protocolos para suprirem novas demandas e solucionarem novos problemas.

A interoperabilidade ainda é problema quando falamos da camada de aplicação. Ainda há um longo caminho para ser percorrido na definição de formatos, consistência e confiança de dados. Outros problemas acontecem também na camada da arquitetura de comunicação entre as aplicações, onde aplicações não seguem à risca a arquitetura de design baseada em RESTful, por exemplo.

Esses problemas não se resolvem da noite para o dia, e não se resolvem sem muito debate e conversa entre todas as pontas: empresas, desenvolvedores, consórcios etc. Todos precisam pensar juntos para que um dia possamos usufruir de um mundo realmente conectado, que troca dados entre si e que, principalmente, transforme a vida das pessoas por meio da tecnologia.

Até lá, o nosso papel, o seu papel, é criar produtos, serviços e websites que adotem os conceitos da Web Semântica e do Linked Data. Aplicar tecnologias como o JSON-LD nos seus sites é realmente um passo bem simples de se dar e, com certeza, a Web ficará mais rica e conectada com a colaboração do mundo.

12.3 Para ler mais

- AGOURAKIS, Dionisio Chiuratto. *Web Semântica e Internet das Coisas*. 2014. Disponível em: http://pt.slideshare.net/DionisioChiurattoAgo/web-semntica-e-internet-das-coisas-gdg-sorocaba-2014
- BARNAGHI, Payam. *Semantic technologies for the Internet of Things.* 2014. Disponível em: http://pt.slideshare.net/PayamBarnaghi/semantic-technologies-for-the-internet-of-things.
- BENSON, Buster. *Cognitive bias cheat sheet.* 2016. Disponível em: https://betterhumans.coach.me/cognitive-bias-cheat-sheet-55a472476b18#.39w41jzhg.
- BENSON, Buster. *Cognitive bias cheat sheet, simplified.* 2017. Disponível em: https://medium.com/thinking-is-hard/4-conundrums-of-intelligence-2ab78d90740f#.qo54o4lei.
- CHUNG, Becky. *How is technology changing our experiences? Further reading on the effects of the "digital now".* 2013.

Disponível em: http://blog.ted.com/how-is-technology-changing-our-experiences-reading/.

- COOPER, Belle Beth. *Novelty and the Brain:* Why New Things Make Us Feel So Good. 2013. Disponível em: http://lifehacker.com/novelty-and-the-brain-why-new-things-make-us-feel-so-g-508983802.
- GITELMAN, Lisa. *Raw Data Is an Oxymoron.* 2013. Disponível em: https://mitpress.mit.edu/books/raw-data-oxymoron.
- KAPLAN, Ken. *Making 5G the do-everything wireless network* 2016. Disponível em: http://iq.intel.com/making-5g-wirelss-networks-the-do-everything-wireless-networks/.
- PARKVALL, Stefan. *Release 14 - The Start of 5G Standardization.* 2015. Disponível em: https://www.ericsson.com/research-blog/lte/release-14-the-start-of-5g-standardization/.